Search Methodologies for Examination Timetabling

Syariza Abdul Rahman, BSc., MSc.

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

July 2012

Abstract

Working with examination timetabling is an extremely challenging task due to the difficulty of finding good quality solutions. Most of the studies in this area rely on improvement techniques to enhance the solution quality after generating an initial solution. Nevertheless, the initial solution generation itself can provide good solution quality even though the ordering strategies often using graph colouring heuristics, are typically quite simple. Indeed, there are examples where some of the produced solutions are better than the ones produced in the literature with an improvement phase. This research concentrates on constructive approaches which are based on squeaky wheel optimisation i.e. the focus is upon finding difficult examinations in their assignment and changing their position in a heuristic ordering. In the first phase, the work is focused on the squeaky wheel optimisation approach where the ordering is permutated in a block of examinations in order to find the best ordering. Heuristics are alternated during the search as each heuristic produces a different value of a heuristic modifier. This strategy could improve the solution quality when a stochastic process is incorporated. Motivated by this first phase, a squeaky wheel optimisation concept is then combined with graph colouring heuristics in a linear form with weights aggregation. The aim is to generalise the constructive approach using information from given heuristics for finding difficult examinations and it works well across tested problems. Each parameter is invoked with a normalisation strategy in order to generalise the specific problem data. In the next phase, the information obtained from the process of building an infeasible timetable is used. The examinations that caused infeasibility are given attention because, logically, they are hard to place in the timetable and so they are treated first. In the adaptive decomposition strategy, the aim is to automatically divide examinations into difficult and easy sets so as to give attention to difficult examinations. Within the easy set, a subset called the boundary set is used to accommodate shuffling strategies to change the given ordering of examinations. Consequently, the graph colouring heuristics are employed on those constructive approaches and it is shown that dynamic ordering is an effective way to permute the ordering. The next research chapter concentrates on the improvement approach where variable neighbourhood search with great deluge algorithm is investigated using various neighbourhood orderings and initialisation strategies. The approach incorporated with a repair mechanism in order to amend some of infeasible assignment and at the same time aiming to improve the solution quality.

Acknowledgements

In the name of Allah, the Beneficent, the Merciful.

Firstly, I would like to express my deepest gratitude to all my supervisors, Prof. Edmund K. Burke, Prof. Andrzej Bargiela, Dr. Ender Özcan and Dr Barry McCollum (Queen's University of Belfast) for their support and assistance through out my PhD journey. I am most fortunate to have the advice and guidance of many talented people, whose knowledge have enhanced this study in so many ways.

I would like to express my sincere gratitude to those who had been involved in contributing their time, effort and support during my time in this university especially to all members of ASAP research group. It is also a pleasure to thank Public Services Department of Malaysia and Universiti Utara Malaysia for their sponsorship.

Finally, I'm forever indebted to my family who has always been understanding and supporting throughout my study. My special thanks to beloved husband Abdul Manaf Othman for his patient, care, love and prayer, and to my children, Ameerul, Marsya, Aida, Aina and Kaisa. Not to forget my special thanks to my parents, my sister and brothers for their constant support and prayer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Timetabling

Timetabling problems can be classified as a type of scheduling problem. A timetable usually provides information about the time for particular events to occur, and eventually relates to the allocation of resources (Wren, 1996). In real world timetabling problems, the allocation of resources at the specified time is required. The task is challenging due to the large number of entities that need to be scheduled and the extensive constraints and preferences that must be satisfied. According to Burke et al. (2004c),

"A timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible."

Much research has been conducted in this area, e.g. sport timetabling (Trick, 2011), vehicle timetabling (Brandão de Oliveira and Vasconcelos, 2010), employee timetabling (Meisels and Schaerf, 2003), educational timetabling, which includes school timetabling (Avella et al., 2007), examination timetabling (Qu et al., 2009b) and course timetabling (Abdullah et al., 2005). Timetabling can be considered as a placement procedure of certain resources to particular events within certain time periods. At the same time, this placement process should satisfy the restrictions related to the objective function of an institution's needs.

Generally, educational timetabling can be categorised into three major groups (Schaerf, 1999) and defined as in Table 1.1. The basic characteristics of these three groups are similar in that they are required to be clash-free i.e. students or teachers cannot attend two or more time-slots at any one time; however there are some significant differences. In school timetabling, students must attend one full day class, while in university course timetabling, the students are required to attend courses which are spread over the whole week; at the same time the students are also given flexibility in choosing courses. With examination timetabling, the requirement is to avoid, if possible, more than one examination in a day for any student. Usually this involves a fixed length of examination duration which occurs twice in a year and which could last for two or three weeks.

Table 1.1: The category of educational timetabling

Category	Descriptions
School timetabling	Scheduling of school classes into time-slots over a week: no teacher or student required to attend two classes at one time.
Course timetabling	Scheduling of courses to time-slots and rooms over a week: no teacher or student required to attend two classes at one time while satisfying room capacity requirement.
Examination timetabling	Scheduling of examinations to limited time-slots (usually the duration is two or three weeks): no student required to sit two or more examinations at one time. In most cases, the room capacity requirement is considered at the same time.

The focus of this thesis is the examination timetabling problem. This problem is considered as an NP hard real world problem (Even et al., 1976). The real world problems are rich and varied, involving significant levels of information from related problems. They have gradually become more challenging in recent years due to the increase in student enrolments and the growing flexibility of course choices (McCollum, 2007). The manual solution of this problem is typically suboptimal (a feasible but not a very satisfactory solution) since the extensive exploration of the solution space is beyond the scope of ad-hoc search. Further research is required to enhance the quality of the obtained timetable in order to satisfy both institutional and personal preferences.

Carter and Laporte (1996) defined the examination timetabling problem as *the assigning of examinations into a limited number of time-slots so that there are no conflicts or clashes.* The key objective of studying the examination timetabling problem is to determine the timetable that optimises some desired objective functions. A set of examinations $E = e_1, e_2, ...e_n$ must be assigned to a limited number of time-slots $T = t_1, t_2, ...t_m$ i.e. it is subject to certain restricted constraints. In assessing the solution to this problem, there are both *hard* and *soft* constraints. The hard constraints must be strictly adhered to under any circumstances, and when satisfied, produce a *feasible* solution. For example, students cannot sit two examinations at the same time. On the other hand, the soft constraints, such as giving students as much free time as possible between examinations, do not affect the feasibility of the solution, although they need to be satisfied as much as possible for the solution to be of high quality. Of course, soft constraints usually have to be violated to some degree in a real world situation. The extent to which the defined soft constraints are satisfied reflects the quality of the obtained timetable.

A survey presented in Burke et al. (1996a) revealed that many different sets of constraints are highlighted by different academic institutions in Britain. The preferences of the constraint evaluation are usually based on the needs of the institution. Such constraints can be categorised as time- and resource-related. Qu et al. (2009b) in assessing timetable quality, described the most used hard and soft constraints within the timetabling community (as shown in Table 1.2 below).

Table 1.2: The most used hard and soft constraints

Constraint type	Descriptions
Hard constraints	- No student should sit two examinations at any one time.
	- The total number of students in the examination room should not exceed the room capacity.
Soft constraints	- Examinations that are in conflict should be distributed within the timetable as evenly as possible.
	- Some examinations are required to be scheduled at a particular location or on the same day.
	- Examinations should be scheduled consecutively.
	- Examinations with large enrolment size should be scheduled as early as possible.
	- Examinations with limited enrolment should be scheduled into a particular time-slot.
	- Some examinations are required to be scheduled within a particular time-slot.
	- Examinations in conflict on the same day should be located nearby.
	- Examinations should be split over similar locations.
	- Examinations with the same duration should be allocated the same room.
	- Resource requirements for certain examinations should be met

Real-world timetabling problems typically require a significant computational effort due to the need to satisfy as many real-life constraints as possible. It is extremely challenging to obtain a good quality timetable using exact methods in many real-world situations and researchers have tended to resort to heuristic approaches. A range of timetabling problems and solution methodologies have been discussed in the academic literature focusing on their complexity and problem solving efficiency, respectively.

The examination timetabling problem is very well known to the timetabling community because of its inherent difficulty. There is almost no known proven optimal solution even for most small real world problem instances. The largest such instance is reported in Parkes and Özcan (2010) as a timetabling problem with thirty-eight examinations to be scheduled at Yeditepe University. As a result, the goal of examination timetabling is frequently defined as finding a high quality (but not necessarily optimal) schedule for a given set of examinations subject to various institutional and personal constraints and preferences. According to Burke et al. (2010b), a constructive approach starts with an empty solution and incrementally builds a complete solution using construction heuristics. The graph colouring heuristics can be used as the construction heuristics within the timetabling problem. On the other hand, the improvement approach starts with a complete solution and is concerned with further refining the current solution quality iteratively, using a set of neighbourhood structures and/or simple local searcher until a stopping condition is met. This thesis focuses on solving the examination timetabling problem using an approach which consists of both constructive and improvement phases.

These problems can, therefore, be mapped through an identity relationship onto the problem of colouring a graph in graph theory. Indeed, this observation underpins some of the earliest and best-known approaches to examination timetabling problems (Carter, 1986). The graph colouring problem is defined as the problem of colouring vertices of a graph with the least number of colours so that no two vertices connected by an edge have the same colour. The vertices represent examinations and the edges connecting vertices represent hard constraints such as student conflict between the examinations. In timetabling, the objective is to schedule examinations in the time-slots while ensuring that the conflicting examinations are not assigned to the same time-slot. The details on the graph colouring problem see Burke et al. (2004c) and the approaches employed to examination timetabling problem can be found in Chapter 2.

A compilation of educational timetabling problems and approaches for solving them can be found in conference volumes by Burke and Ross (1996), Burke and Carter (1998), Burke and Erben (2001), Burke and De Causmaecker (2003), Burke and Trick (2005), Burke and Rudová (2007) and Burke and Barry (2010).

1.2 Research Motivation

The successful assignment of an examination to a time-slot can be strongly dependent on the order in which examinations are processed. Consequently, an investigation of the examination ordering strategy is an important part of this thesis. In particular, the present research investigates the ordering of examinations according to the perceived difficulty of scheduling them in the available time-slots. The examinations deemed to be the most difficult are scheduled first in the timetable in the hope that the remaining scheduling problem would be less difficult than the original task. The relatively less difficult examinations are assigned at the later stages in the timetabling process. The approaches to timetabling which encompass this basic graph colouring implementation are known as constructive approaches and are often used during the initialisation strategy before going on to the improvement phase.

Since none of the ordering strategies provides a guarantee of successful scheduling, there have been extensive studies on constructive approaches reported in academic literature. Most of these incorporated graph colouring heuristics have employed some adaptive strategies (Abdul Rahman et al., 2009, Burke and Newall, 2004), fuzzy techniques (Asmuni et al., 2009, Pais and Burke, 2010), decomposition (Abdul Rahman et al., 2010, Qu and Burke, 2007), granular modelling (Abdul Rahim et al., 2009) and neural networks (Corr et al., 2006). As a result, approaches related with these ordering strategies are used either for constructing a good quality timetable or for producing a good initial solution before proceeding to improve the solution quality. As shown by several studies, a good initial solution can help to produce better final solutions (Burke and Newall, 2003, Gogos et al., 2010a). These findings have motivated this research to focus on searching for good quality solutions during the timetable construction and, additionally, the solutions are improved in the improvement phase. In examination timetabling, this phase is typically concerned with improving the solution quality using other sophisticated approaches such as meta-heuristics.

The research begins with an investigation into the constructive approaches that are based on the ordering strategy from the graph colouring heuristics. An investigation of the squeaky wheel optimisation and decomposition strategies seeks to change the ordering by referring to the unscheduled examinations encountered in previous timetable constructions. In addition, an improvement approach is also considered in order to improve the generated solution in various ways, using a variable neighbourhood search - great deluge algorithm where the focus is to investigate the influence of different initialisations and neighbourhood orderings on producing good quality solutions.

1.3 Research Objective

The research into constructive and improvement approaches examined two different examination timetabling problems, known as the Toronto instances and the Second International Timetabling Competition (ITC2007) benchmark datasets. In the first phase, the study concentrates on finding good quality solutions based on an ordering strategy where the heuristics used are adapted from graph colouring heuristics. The squeaky wheel optimisation is simultaneously explored in changing the examination ordering, while in other research work, considers a decomposition strategy in order to divide the problem based on the difficulty of scheduling individual examinations. The strategies investigate the unscheduled examinations that are obtained in the previous timetable construction by giving them priority to be scheduled first. In the second phase, an improvement strategy is employed to refine the previously identified solutions. The objectives listed below summarise the scientific aim of the thesis. Objectives one to three focus on the construction of an examination timetable while objective four seeks on the improvement of the solution quality:

1. **An investigation of the squeaky wheel optimisation strategy for examination ordering**: this research identifies the effect of changing the examination ordering within the block or top-window strategy and the use of heuristic modifier and graph colouring heuristics that contribute to the solution quality and the number of unscheduled examinations during timetable construction. Further, the choice of time-slot selection, shuffling current best ordering whenever no improvement for a certain period and the combination of a number of graph colouring heuristics while constructing the solutions are considered throughout the timetabling process. The research also aims at contributing to an improved understanding of how various types of graph colouring heuristics provide a different number of unscheduled examinations when incorporated with different types of heuristic modifier.

2. **A further investigation of the squeaky wheel optimisation concept with the linear combination of heuristics**: this research investigates the advantages of combining more than one graph colouring heuristic simultaneously with a heuristic modifier in linear approach in order to find the best ordering for examinations based on examination difficulty. Using this method, each graph colouring heuristic and heuristic modifier is invoked with a normalisation strategy in order to generalise the specific problem data and is embedded with a different combination of weights values. The research aims at obtaining a new score value for each examination based on the difficulty of assigning examinations to time-slots. This new score value is used for ordering and constructing examination timetables. Instead of fixing the weight value for each parameter, the study also seeks to investigate the changes of weight values automatically while constructing the examination timetables.

3. **A development of decomposition strategy that makes use of information obtained from the unscheduled examinations**: the problem is decomposed

into two subsets and each subset is embedded with a heuristic ordering and shuffling strategy with the roulette wheel selection method. The research issue is to identify whether the examinations that generate an infeasible timetable can generate good initial solutions when they are treated first. A new subset, known as a boundary set, is introduced in order to vary the examination orderings.

4. **An investigation of variable neighbourhood search - great deluge algorithm to improve the solution quality from the constructed solutions**: the research aims to build an algorithm that can effectively improve the solution quality constructed using constructive approaches that are introduced. Different neighbourhood structures are considered, including the Kempe-chain moves that accept infeasible moves and this infeasibility is resolved using a repair mecahnism. The study is concerned with testing different neighbourhood orderings and initialisations in order to observe the effect on the solution quality obtained.

1.4 Research Contribution

The research reported in this thesis resulted in a number of original research contributions as described below:

1. An investigation into adaptive heuristic ordering based on the squeaky wheel optimisation has been incorporated with a shuffling strategy. This has shown an improvement to the solution quality when compared with the ordering without the shuffling strategy. Statistical analysis revealed that different sizes of block or top-window strategy could significantly affect the solution quality. The study found that different graph colouring heuristics employed in the approach give a different number of unscheduled examinations at each iteration. It is also demonstrated that the incorporation of different types of heuristic modifier has a significant effect on the contribution of the number of unscheduled examinations at each iteration and at the same time greatly influences the solution quality. In addition, the study has established that the use of more than one graph colouring heuristic while constructing the solution is beneficial where a number of graph colouring heuristics were alternated during the timetable construction, and that this could assist in obtaining a good quality timetable.

2. Within the squeaky wheel optimisation method, the combination of a number of graph colouring heuristics with a heuristic modifier adapted with weight values within a linear approach can significantly improve the examination ordering where the ordering is based on a new difficulty score value. This approach has led to good

quality timetables. The research concluded that different weight values adapted to different parameters in the linear approach could influence the solution quality. Moreover, instead of using a fixed weight value for each parameter, the automatic weight changes during timetable construction have the potential to produce a good examination ordering.

3. An investigation of the decomposition approach to the unscheduled examinations showed that scheduling the difficult examinations first could be advantageous for improving the solution quality. The study introduced a boundary set that is located between the difficult and easy set. Merging or swapping the boundary set with the difficult set has produced improved solution quality. Furthermore, a stochastic component based on a roulette wheel selection was embedded into the approach in order to enhance the probability of selection of an examination with a high score of difficulty. It is concluded that this decomposition strategy approach, which gives priority to the unscheduled examinations obtained in previous iterations, could enhance the solution quality obtained.

4. The research has ascertained that the improvement approach based on variable neighbourhood search and the great deluge algorithm is competitive when compared with other improvement approaches in the literature. The approach accepts infeasible moves in order to diversify the search. In order to overcome the infeasibility, a repair mechanism was introduced which removed the infeasibility and, at the same time, further improved the solution quality. The finding of this research is that the neighbourhood orderings and initialisations give an advantage for producing good solution quality and these have an effect depending on the characteristic of the implemented problems.

1.5 Dissemination

A number of research results described in this thesis have been published or are accepted for publication in peer-reviewed publication outlets. Additionally, there are papers currently being prepared for submission for publication. The following is a list of the papers that have emerged from this thesis:

1. Syariza Abdul Rahman, Andrzej Bargiela, Edmund K. Burke, Barry McCollum, and Ender Ozcan. Construction of examination timetables based on ordering heuristics. In Proceedings of the 24th International Symposium on Computer and Information Sciences, pages 727-732, 2009.

2. Syariza Abdul Rahman, Andrzej Bargiela, Edmund K. Burke, Barry McCollum, and Ender Ozcan. A construction approach for examination timetabling based on adaptive decomposition and ordering. In Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland, pages 353-372, 2010.

3. Syariza Abdul Rahman, Andrzej Bargiela, Edmund K. Burke, Barry McCollum, and Ender Ozcan. A constructive approach for examination timetabling based on adaptive decomposition and ordering. Accepted for publication in Annals of Operations Research.

4. Syariza Abdul Rahman, Andrzej Bargiela, Edmund K. Burke, Barry McCollum, and Ender Ozcan. Linear combination of adaptive heuristic orderings in constructing examination timetable. Submitted to European Journal of Operational Research.

5. Syariza Abdul Rahman, Andrzej Bargiela, Edmund K. Burke, Barry McCollum, and Ender Ozcan. Initialisation and neighbourhood ordering in VNS for improving examination timetables. In preparation for submission to Journal of Scheduling.

1.6 Outline of the Thesis

This thesis comprises seven chapters. The first chapter has presented an introduction of the timetabling problem as well as the motivation and aims of the research and an outline of the dissemination of the research work. Chapter 2 describes recent approaches to the examination timetabling problem and categorises them into several major groups. The benchmark datasets widely used in the examination timetabling community are also described in this chapter.

Chapters 3, 4, 5 and 6 introduce new approaches to the examination timetabling problem and discuss their implementation and the analysis and experiments for the two well-known benchmark datasets. Chapter 3 is an initial investigation of the squeaky wheel optimisation strategy introduced by Joslin and Clements (1999) and first implemented in examination timetabling by Burke and Newall (2004). The unscheduled examinations are identified as difficult and their difficulty levels are increased so that they are given priority to be chosen first in the next iteration. The investigation focuses on the examination selection in the ordering where the change of position of ordering within the block or top-window strategy are identified. At the same time, shuffling current best ordering whenever no improvement to the solution quality for a certain period and alternating a number of graph colouring heuristics in order to change examination ordering are also explored, while the choice of time-slot selection is examined throughout the timetabling process. Further, the chapter presents an investigation into the contribution of the number of unscheduled examinations produced when using different graph colouring heuristics and different heuristic modifiers.

The squeaky wheel optimisation technique is further explored in Chapter 4 where the concept of a linear approach that combines a number of graph colouring heuristics with a heuristic modifier is studied. This research investigates the merits of combining more than one graph colouring heuristic in order to find the best ordering for the examinations based on their difficulty. A normalisation strategy is invoked for each heuristic and is embedded with different combinations of weights values. The research generates a new score value for each examination and a new ordering of examinations is obtained based on the difficulty of assigning examinations to the time-slots. Several strategies on changing the weights automatically are also examined.

In Chapter 5, a further study of unscheduled examinations considers the decomposition of the problem into two subsets that divide unscheduled and scheduled examinations so that they can be timetabled separately. The unscheduled subsets of examinations are given priority by scheduling them first, followed by the scheduled subset. A fixed size of boundary set is introduced in the chapter in order to diversify the search, and a number of experiments are performed involving different heuristics used for ordering the examinations in each set.

In Chapter 6, a variable neighbourhood search - great deluge algorithm is employed on the examination timetabling problems in order to improve the solution quality yielded by the constructed solutions in previous chapters. The approach is incorporated with a mechanism that repairs the infeasible examination moves while at the same time aiming for further improvement of the solution quality. Different neighbourhood orderings and initialisations are investigated in order to observe their effect on the solution quality.

Finally, Chapter 7 offers overall conclusions drawn from the present study, and outlines the research direction for future work based on the ideas and results presented in this thesis.

Chapter 2

A Survey of Algorithmic Approaches for Examination Timetabling

This chapter overviews the existing approaches to examination timetabling. Section 2.1 points out the surveys on examination timetabling, while Section 2.2 discusses the solution methodologies used for examination timetabling in six different categories. In Section 2.3, a survey on the approaches proposed for solving the real world problem instances from the second International Timetabling Competition (ITC2007) is presented. Section 2.4 introduces the real world examination timetabling problem datasets in the literature. Section 2.5 describes the benchmark problems used during the study in detail, covering both hard and soft constraints along with the relevant evaluation functions. Finally, Section 2.6 provides a summary.

2.1 Survey of Examination Timetabling

The examination timetabling problem has been extensively studied and a wide range of approaches has been taken across a variety of associated problem descriptions. This problem is very well recognised in the academic literature and surveys have concentrated on various methods and strategies for solving it (Carter, 1986, Carter and Laporte, 1996, Lewis, 2008, Qu et al., 2009b, Schaerf, 1999). One of the earliest surveys was conducted by Schmidt and Strohlein (1980), who presented studies related to examination timetable constructions which appeared in 1979 or earlier. Most of the review studies at that time presented the timetabling problem as a graph colouring problem.

A later survey by Carter (1986), one of the most popular timetabling surveys within the timetabling community, discussed major approaches to examination timetabling, and modeled the problem as a graph colouring problem. The survey, carried out from 1964 to 1984, was a chronological discussion of the implementation of graph colouring heuristics and the requirement of handling secondary constraints in several institutions. At that time, the graph colouring problem was used to find a non-conflict timetable, and there was no attempt at hybridising the approach or to find the best approach for testing the problem.

As a continuation of the first survey paper in 1986, Carter and Laporte (1996) continued to contribute to the examination timetabling literature. The focus of the survey is the research of examination timetabling approaches in the 1990's. Carter and Laporte (1996) provided a general definition of the examination timetabling problem. Several common soft constraints were discovered, some of these being associated with related resources such as room availability, requirement of special resources, invigilators and examiners' requirements. The authors introduced public datasets which are used widely today as a test bed for the examination timetabling problem and which are referred to as the 'Toronto' sets. The first implementation of these datasets was by Carter et al. (1996), a study which has encouraged researchers within the timetabling community to engage in research of this problem in order to improve the solution quality and at the same time give new insight into timetabling approaches. In their paper, Carter and Laporte (1996) categorised examination timetabling into four main methods: cluster, sequential, meta-heuristic and constraint-based. Their surveys illustrated several successful implementations of these methods.

In the same year, Bardadym (1996) presented a survey on educational timetabling problems and discussed automated timetabling approaches. Educational timetabling was classified into five common types, namely, faculty timetabling, class-teacher timetabling, course timetabling, examination timetabling and classroom assignment. The university timetabling problem was said to be the most difficult task in educational routine work. The difficulties arise when the problem is large, the requirements are contradictory and there is an interaction of various constraints in finding a better quality timetable. According to Bardadym (1996), work on timetabling systems was first attempted when computers became available in universities. Early approaches to timetabling were based mainly on mathematical programming and combinatorial approaches (which included representing the timetabling problem as a graph colouring problem) as well as on network flows and transportation problems. Some other combinatorial structures, such as integer mathematical programming and general scheduling theory (network planning and calendar scheduling) were also included.

In an attempt to identify various constraints related to university timetabling, Burke et al. (1996a) conducted a survey on examination timetabling in fifty-six universities in Britain, which revealed that a wide range of constraints had been used. Consequently, various constraints and the needs of different institutions were found very significantly across institutions.

In a previous study, Schaerf (1999) grouped the educational timetabling problem into three categories: school, course and examination, all sharing similar basic characteristics. Discussions were focused on the mathematical formulations, variants and techniques that had been applied to the three problems. The approaches implemented to the examination timetabling problem consisted of direct heuristics, graph colouring, simulated annealing, genetic algorithms and other techniques related to a network model with a Lagrangian relaxation.

Burke and Petrovic (2002) discussed three major approaches in educational timetabling, namely, heuristic and evolutionary algorithm, multi-criteria decision making and case-base reasoning. Their study revealed that interest in educational timetabling at that time focused on the meta-heuristic and hybrid approaches. As described by Burke and Petrovic (2002), a solution generated using a constructive heuristic alone was not enough to provide a good solution quality. These approaches concentrated on improving the solutions by employing a search strategy to avoid getting stuck in the local optima; however, they needed a proper parameter setting and involved some computational cost. On the other hand, the multi-criteria approach could handle many criteria simultaneously at any one time, while the case-base reasoning used the previous problems, storing the solutions in the case base. Both approaches showed significant achievement in the timetabling arena.

An overview of state of the art methods dealing in timetabling problems was presented by a research group led by Petrovic and Burke (2004). These consisted of meta-heuristic and multi-criteria approaches, case-based reasoning and hyper-heuristics, and self-adaptive approaches. Meta-heuristics were successfully applied to the university timetabling during the last two decades. Nevertheless, these approaches require certain good parameter tuning in order to produce good solution quality. As suggested by Petrovic and Burke (2004), the incorporation of parameters that can be understood by users of the approaches could produce worthwhile real-world implementations. Moreover, the meta-heuristic approaches presented in their study were less dependent on parameter setting. In the application of case-based reasoning approaches, Petrovic and Burke (2004) employed previous knowledge of timetabling by representing it within two important roles, namely, 'solution reuse' and 'methodology reuse'.

The success of the meta-heuristic approaches has given attention to Lewis (2008) to classify them into three groups that were related to the hard and soft constraints, which are one-stage optimisation algorithms, two-stage optimisation algorithms and algorithms that allow relaxations. The one-stage optimisation algorithms satisfied both hard and soft constraints simultaneously and the violations are allowed in order to achieve good quality solutions, while the two-stage optimisation algorithms focused on satisfying the soft constraints as much as possible once the feasible timetable is obtained. On the other hand, the algorithms that allow relaxations employed a relaxation strategy to the implemented algorithms in order to give more possibility of satisfying the soft constraints and at the same time maintained the feasibility of the problems. These classifications were followed by some illustrations of previous implemented approaches.

In the most recent survey of examination timetabling problems, Qu et al. (2009b) provided an extensive survey on the development of search methodologies and automated systems for examination timetabling. This survey was a much updated discussion of the literature which has expanded considerably since the earlier study by Carter and Laporte (1996) on algorithmic approaches. Qu et al. (2009b) highlighted the new research pattern in the timetabling problem and the achievement in finding solutions in the last decade. The survey found that meta-heuristic approaches and their hybridisation with other search techniques have been implemented quite extensively. Furthermore, Qu et al. (2009b) drew attention to some clarification issues concerning the naming of the examination timetabling problem by the benchmark datasets. Some of the benchmark datasets used by researchers were discovered to be slightly different (but having the same name). This has created great confusion among researchers in creating various solutions from diverse benchmark datasets. Thus, an alternative naming convention was proposed by Qu et al. (2009b) to differentiate those datasets in order to avoid future misunderstanding. The survey paper also listed some automated systems and their development in the timetabling area.

2.2 Algorithmic Techniques

Most surveys in examination timetabling problems place the algorithmic techniques into several categories (Burke and Petrovic, 2002, Carter and Laporte, 1996, Lewis, 2008, Petrovic and Burke, 2004, Qu et al., 2009b, Schaerf, 1999). The following discussion divides these algorithmic techniques into subsections each considering one of six categories: exact, constraint-based, heuristic, meta-heuristic, hyper-heuristic and multi-objective approaches. The meta-heuristic technique comprises two major methodologies i.e. local search-based methodologies and population-based methodologies. The hyper-heuristic techniques can be further divided into two categories with respect to the nature of the heuristic space during the search process at a specific decision point, i.e. heuristic selection methodologies that deal with the existing heuristics and the combinations, and heuristic generation methodologies that generate and employ new heuristic methods from the existing heuristics. Further, the variable neighbourhood search and large neighbourhood search as meta-heuristic approaches can also be classified as hyper-heuristic when the heuristics used are chosen by the high level heuristic. Most of the existing algorithmic techniques applied to timetabling are based on single objective models; however, there are a few studies which concentrate on the multi-objective and multi-criteria approaches in the literature. Other recent approaches related to timetabling are fuzzy-based, granular modeling, developmental approach and harmony search algorithm.

2.2.1 Exact Approaches

Exact approaches represent a classical search methodology that evokes mathematical procedures. Usually, mathematical formulations are incorporated in order to represent the objective of solving problems and constraint requirements. Examples of these approaches include integer programming (Bosch and Trick, 2005, Sierksma, 2001), linear programming (Sierksma, 2001) and branch and bound (Chen and Bushnell, 1996). Currently, their application to timetabling problems is being investigated in order to solve extensive difficulties. Such approaches aim to find an optimal solution for a particular issue. However, they are not always successful, especially when attempting to solve widespread timetabling problems due to computational expense. Furthermore, the mathematical model of these approaches needs to be carefully developed and treated.

In order to solve the examination timetabling problem at the University of Technology of Compiègne, Boufflet and Nègre (1996) used an exact approach and developed three methods by considering several different constraints. The exact approach based on the graph coloring technique was used to search the path in the tree. These were combined with tabu search and an interactive computer-aided design system. The tree search as an exact approach was shown to perform well in solving the problem. However, the approach has never been applied to any benchmark problem and its effectiveness cannot be assessed and compared with other approaches in the literature.

A recent study by MirHassani (2006) modelled the examination timetabling problem at the Shahrood University of Technology as an integer programming approach. The objective of the study was to maximise the examination spread in the timetable. The study presented a mathematical model formulation and considered three types of examination clashing, i.e. in the same time-slot, on the same day and on two consecutive days. The approach was able to produce a conflict-free timetable within a reasonable running time. Meanwhile, Qu et al. (2009c) investigated the application of the integer programming approach in solving the difficult sub-problem of examination timetabling. The nature of the problem at first was divided into 'easy' and 'difficult', as described by Qu and Burke (2007). The difficult sub-problem was then solved using an integer programming model in order to achieve an optimal solution as this sub-problem was said to add a large amount of cost even though the size of the problem was small. The integer programming model introduced in the study was adapted with clique inequalities. Moreover, the problem of specific cutting planes was introduced in order to reduce the gap to the optimal solution to less than 10% from the constructed solutions. This integer programming approach, hybridised with a decomposition strategy, has achieved promising results regarding the tested problems.

2.2.2 Constraint-based Approaches

Examination timetabling can employ a constraint-based approach. The survey by Carter and Laporte (1996) highlighted the constraint-based approach which included constraint logic programming and constraint satisfaction problems. In a study, Merlot et al. (2003) presented a three-phase hybrid algorithm for capacitated and un-capacitated examination timetabling problems. The study hybridised a three-phase approach that makes use of constraint programming, simulated annealing and hill climbing. The constraint programming employed in the approach was similar to that used by Boizumault et al. (1996). The first phase used constraint programming in order to create a feasible solution in a very short time. If during the process the examination still could not be scheduled, then the algorithm allowed the examination to remain unscheduled. Once more, the solution was enhanced in the improvement phase where simulated annealing with the Kempe-chain neighbourhood (Thompson and Dowsland, 1996a) was employed to allow diversification in the search. The solution then was further improved by the hill climbing method. This hybridisation technique was tested on the problem faced at the University of Melbourne, and proved to be superior to the previous method being applied there, although testing with datasets at Toronto and Nottingham have provided the best results.

A study by Duong and Lam (2004) employed a constraint programming approach to construct an initial feasible timetable before improving it with the simulated annealing technique. The constraint programming introduced in the study used a 'backtracking with forward checking' (BC-FC) strategy so as to obtain a good initial solution. The reason was that the success of the simulated annealing approach was vitally determined by the quality of the initial solution that was fed into it. The BC-FC strategy performed a dynamic variable ordering that determined the priority of each examination. The examinations were ordered based on the priority score value that were introduced in the study before assignment to the available time-slots was made.

2.2.3 Constructive Heuristic Techniques

2.2.3.1 Graph-based Heuristics

The examination timetabling problem can be represented as a graph colouring problem where the vertices represent the examinations, the edges represent the conflict between two examinations and the colour of the vertices represent different time-slots in the timetable. The discussion of the timetabling problem as a graph theoretical model is described in studies by de Werra (1985), de Werra (1997) and Burke et al. (2004c).

A definition of the concepts and terms that relate to a graph can be found in Burke et al. (2004c). An undirected graph $G = (V, E)$ is a representation that consists of a set of vertices, $V = v_1, ..., v_n$, and a set of edges, E. If (v_i, v_j) is an edge in a graph $G = (V, E)$, then vertex v_i is adjacent to vertex v_j (Burke et al., 2004c). The graph colouring method creates a timetable by using a sequential strategy. Most of the early timetabling studies used the sequential technique to solve the problem because it was the simplest and easiest to implement. It is based on an ordering strategy which allows the examination with the most difficulty to be chosen first, by trying to place the examinations sequentially into time-slots in order to produce a timetable. Burke et al. (2004c) also listed the most commonly used graph colouring heuristics for examination timetabling, i.e. largest degree, largest weighted degree, colour degree and saturation degree. From this point onwards, the graph-based ordering strategies in this thesis are referred to as graph colouring heuristics.

A study by Broder (1964) was one of the earliest that utilised the ordering strategy of graph colouring theory for solving examination timetabling problems. The use of the ‘largest degree’ heuristic as an ordering strategy was based on the difficulty of the examination to be scheduled. This difficulty was measured by the number of edges connecting to vertices. The largest number of edges of a vertex shows which examination is the most difficult to schedule due to the large number of conflicts with other examinations.

A study by Cole (1964) also employed the largest degree heuristic to examination timetabling problems. A table of conflict matrix $N \times N$ was presented which listed the conflict by courses and in order to allocate the courses into suitable time-slots, the largest degree heuristic was employed. Further, Peck and Williams (1966) presented the graph colouring procedure when using the largest degree heuristic for examination timetabling. The ordering was modified by partitioning and rearranging the assignment of examinations to time-slots. Welsh and Powell (1967) also made use of the graph colouring heuristic in order to find the least number of colours (*chromatic number*) applied to the vertices of a graph. The main idea was to create a non-conflict graph where there was no matching colour for two adjacent vertices. The largest degree heuristic was used in identifying the colour of the vertices.

The relationship between the examination timetabling and the graph colouring problem was studied by Wood (1969) who identified the chromatic number of a graph so that the number of time-slots needed in a certain examination session could be obtained. In another study, Wood (1968) constructed a system for university timetabling for the University of Manchester. In order to determine examinations to be fitted into timetables, an ordering strategy called 'largest enrolment' was introduced. This strategy attempted to fit the examinations with the largest student enrolment in the first time-slot in the timetable. Three different methods of identifying the examination session of the shortest duration were compared, i.e. largest degree heuristic, similarity matrix and the upper bound approach proposed by Welsh and Powell (1967). In this study, the similarity matrix produced the best result among them. However, the approach was tested on a small problem involving only twenty vertices in the graph colouring problem.

A dynamic ordering strategy, known as 'saturation degree', was first introduced by Brélaz (1979). This method is very effective compared with other heuristic methods because it dynamically colours the vertex. This heuristic has been applied successfully to examination timetabling (Abdul Rahman et al., 2009, Burke and Newall, 2004, Carter and Laporte, 1996). It works by giving priority to the vertex with the least colour available to be coloured first. Practically, the vertex with the least available colour is the most difficult to be scheduled since it has the smallest saturation degree. A study by Mehta (1981) was one of the earliest investigations implementing this dynamic ordering strategy to solve the examination timetabling problem. The examinations were ordered based on the number of time-slots in conflict, the examinations with the highest conflicting time-slot being the first to be fitted into the schedule.

Laporte and Desroches (1984) developed a system named HORHEC for solving examination timetabling problems at the University of Montreal Business School. During the construction phase, the solution used several graph colouring heuristics: largest degree, largest weighted degree, largest enrolment and random ordering. This was repeated several times in order to obtain a feasible examination timetable. Whenever there were examinations that could not be assigned into the timetable, a backtracking procedure was incorporated in order to make sure that the problematic examinations could be fitted in.

In another study, Burke et al. (1994) produced a university timetabling spreadsheet type system for examination and course timetabling problems based on graph colouring and constraint manipulation. The events (examination/course) were scheduled in the appropriate time-slot and at the same time fulfilled the room capacity requirement. The user could interact with the system to obtain a desired solution and if no solution was found, the backtracking strategy of the system was employed in order to develop a complete timetable and to improve the solution quality. The examination timetabling system applied to the University of Nottingham dataset with real world features was presented in Burke et al. (1993).

Johnson (1990) constructed examination timetables based on the 'difficulty factor' obtained from graph colouring heuristics. During the first phase of the implemented approach, the combination of largest enrolment and largest degree was used as an ordering strategy to assign examinations to time-slots. Several variations of the relative weights of each criterion were considered in order to produce a number of different timetables. In the next phase, the simulated annealing approach was used to improve the solution quality of the obtained timetable.

Essentially, the sequential heuristics have proved to be very efficient when incorporating a backtracking procedure (Carter et al., 1994). This procedure is implemented whenever some examinations (violated examinations) cannot not be assigned to a timetable due to conflict with other examinations and they should therefore be rescheduled into the available time-slots. Examinations that conflict with the current infeasible examination are unscheduled in order to allow the current violated examination to be scheduled first. Then, the unscheduled examinations are rescheduled back into new time-slots. There are a number of studies relating to the backtracking procedure implemented with graph colouring heuristics in order to obtain a feasible timetable (Asmuni et al., 2009, Burke and Newall, 1999, Carter et al., 1996, Gogos et al., 2008, Laporte and Desroches, 1984).

In 1996, Carter et al. developed a commercial software for examination timetabling, named EXAMINE. Their study incorporated a backtracking procedure in order to resolve the infeasibility problem during the timetable construction. The approach was found to reduce the length of the examination session by half compared with sequential techniques without backtracking. Five graph colouring heuristics were implemented to order the examinations based on its scheduling difficulties (see Table 2.1). The study found that the saturation degree heuristic provided a good sequence ordering of examinations, while experimental results on the Toronto benchmark datasets showed that the quality of the final solution was enhanced when combining a backtracking strategy and tabu list.

Table 2.1: The graph colouring heuristics in examination timetabling

Heuristic	Descriptions
Largest degree (Broder, 1964)	The largest number of edges/conflicting examinations is scheduled first.
Saturation degree (Brélaz, 1979)	Ordering is based on the number of time-slots in conflict where the examination with the fewest time-slots is scheduled first.
Largest weighted degree (Carter and Laporte, 1996)	The examination with the largest number of students who are involved in the conflict is scheduled first.
Largest enrollment (Wood, 1969)	The largest number of students registered for the examinations is scheduled first.
Random ordering	The ordering is random for the purpose of benchmarking and comparison with other sequencing strategies.

Current research trends in examination timetabling include the use of an adaptive ordering technique combined with graph colouring heuristics. The adaptive approach in Burke and Newall (2004) was based on the concept of 'squeaky wheel optimisation', proposed by Joslin and Clements (1999). The approach could adapt to any given problem by adding a *heuristic modifier* to the basic graph colouring heuristic technique and by promoting difficult examinations to be scheduled first at each iteration based on its order. The study took into account different considerations of hard and soft constraints in order to test the effectiveness of heuristic modifiers. The details of this approach are discussed in Chapter 3. The technique introduced a good initialisation strategy for examination timetabling problems, the results demonstrating that the adaptive heuristic ordering approach could improve the quality of the obtained solution compared with using only a basic graph colouring heuristic approach.

The adaptive ordering strategy proposed by Burke and Newall (2004) was further studied by Abdul Rahman et al. (2009) who incorporated shuffling strategies, i.e. block and top-window, to different graph coloring heuristics. These strategies acted as a stochastic component and ordered the examinations within a group of examinations. Different sizes of examinations in a group and diverse modifier types have been tested with the Toronto benchmark datasets. Their study found that the saturation degree heuristic produced better results compared with the largest degree heuristic. It was concluded that the dynamic nature of saturation degree caused this heuristic to work very effectively on the tested problem. For more details on the study see Chapter 3.

A recent study by Carrington et al. (2007) used the weighted graph model proposed by Kiaer and Yellen (1992) for solving examination/course timetabling. The vertices and edges of a graph usually hold much information related to the objective function that help in the ordering of examinations. The weighted graph model was enhanced by introducing several new heuristics for vertex-selection and time-slot-selection and the implementation was varied with various combinations and partitions. The vertex introduced by the graph was selected and coloured until it was finished. The approach was tested on the Toronto benchmark datasets and showed promising results compared with the pure graph colouring heuristics. The study was extended by Burke et al. (2010e) where the introduced heuristics were combined with a linear approach. Moreover, weights were adapted to each heuristic that contributed to the ordering of the vertices.

Kahar and Kendall (2010) solved the examination timetabling problem at the Universiti Malaysia Pahang, using four graph colouring heuristics, the time-slots being chosen based on the best time-slot from the candidate list of five. The approach was able to produce a feasible solution to the problem and was found to be better than the university's current software.

2.2.3.2 Fuzzy-based Techniques

The fuzzy-based technique is underpinned by the concept of fuzzy logic that was first introduced by Zadeh (1965). Essentially, a fuzzy technique deals with the uncertainty condition and relates reasoning with linguistic terms. Typically, it is an approximate approach that can handle several reasons or factors simultaneously in making a decision. Fuzzy-based techniques have now become a successful method in various scheduling areas. An implementation of a fuzzy technique in examination timetabling was carried out by Asmuni et al. (2005), who combined several sets of two graph colouring heuristics and ordered the examinations based on their difficulties. Three graph colouring heuristics were used in the experiment, i.e. largest degree, saturation degree and largest enrolment with three combinations of two heuristics. Their study used the fuzzy approach to represent the knowledge from the heuristics (named as 'input variables'), evaluate them and construct an examination weight as an input variable. The examinations then were placed in a decreasing order based on the examination weight values and scheduled in the timetable without violating any of the hard constraints. The 'bumped back' strategy was employed only if infeasible examinations occurred. The work showed that a tuning procedure was needed for different combinations of heuristics in order to obtain good solution quality.

Asmuni et al. (2009) continued to carry out research on the fuzzy methodology for solving university examination timetabling problems by focusing on the construction phase. The performance of the approach was based on three criteria, namely, the penalty cost compared with other constructive approaches, the number of bump-back strategies required for each dataset and the processing time for a static and dynamic heuristic for each combination. The study obtained one best result on the Toronto benchmark datasets when compared with other constructive approaches at that time.

Pais and Burke (2010) used the fuzzy approach to measure the difficulty of examinations using graph colouring heuristic combined with Choquet integral. In the study, the Choquet integral was used to measure the expected utility of an uncertain event. The study incorporated weight value to each combination and the weight values were identified based on the information of a given graph colouring heuristic. The examination with the highest value of Choquet integral was scheduled first.

2.2.3.3 Decomposition Techniques

Complex problems are often difficult to solve due to the size of search spaces, and the decomposition technique is therefore an alternative to resolving this problem: it is divided into smaller sub-problems that are easier to handle and at the same time give rise to high quality solutions. Nevertheless, feasibility often could not be attained due to early assignment of certain sub-problems (Qu et al., 2009b).

There are a few studies within the field of examination timetabling that are closely related to the decomposition technique. A study by Burke and Newall (1999) implemented the memetic algorithm with a decomposition strategy for the examination timetabling problem. Before proceeding to the next sub-component, this multi-stage approach split the large problems into a defined number of sub-components so that the algorithm could schedule a group of examinations into the timetable at one time. The backtracking and forward checking strategies were incorporated if an infeasible timetable occurred in the early assignment. It was shown that, this algorithm could reduce significantly the processing time and improved the solution quality when tested on four large benchmark datasets.

An investigation into the clique initialisation in examination timetabling problems by Carter and Johnson (2001) observed that more than one maximum clique was available on the tested examination timetabling problem. In graph theory, an undirected graph forms a clique whenever a subset of vertices are all connected by edges. In their study, several dense sub-graphs were found to be potentially almost clique. The examinations in the dense sub-graph were identified as very difficult to solve and should be scheduled earlier.

An adaptive decomposition approach for constructing examination timetables was studied by Qu and Burke (2007). The problem was grouped into two sets, i.e. ‘difficult’ and ‘easy’. The difficult set consisted of difficult examinations that were identified according to the feasibility of the examination in a previous iteration and the size of the set was modified accordingly. At the end of the iteration, the rigid size of the difficult set was determined. On the other hand, the easy set was then reordered so as to improve the solution quality. The study also introduced the concept of a set of ‘boundary examinations’ between the difficult and easy sets, i.e. the difficult and easy sets share some overlapping examinations, which contributed an improved solution quality. The study found that the examinations in the difficult set could contribute a large amount of penalty cost even though the size of the set was small. The approach was tested on the Toronto benchmark datasets and found to be simple and effective.

In another study, Kendall and Li (2008) investigated the simplification of examination timetabling problems by combining several examinations into one examination that had equal features when measuring the compatibility value. The reason for combining those examinations was to reduce the search space during the attempts to construct the timetable. The study has shown that the strategy could increase the solution quality, although it could also contribute to an increase in searching time. So far, this approach has been tested only on the sta83 I of the Toronto benchmark datasets, but it has shown promising results. However, the compatibility measure was hard to find and it does not exist in all problem instances.

When Abdul Rahman et al. (2010) investigated the decomposition of the examination timetabling problem, they stratified it into two sets, as introduced in Qu and Burke (2007). The examinations causing the infeasibility of a solution were moved to the difficult set because this indicated that those examinations were difficult to place and should perhaps have been treated in different ways. Initially, the assumption that all examinations can be easily scheduled presented the problem, and the difficult set was gradually to increase as infeasibility occurred during the timetabling process. The study introduced a boundary set located between the difficult and easy sets, where the examinations in the boundary set could also be considered as difficult. The roulette wheel selection considering information about two graph colouring heuristics was taken into account within a predefined size of top-window in order to shuffle the ordering. For further information on this approach see Chapter 5.

2.2.3.4 The Granular Modelling Technique

One of the latest approaches applied to examination timetabling is the granular modelling technique. This approach focuses on identifying higher level of information entities that simplify the description of the original problem. Abdul Rahim et al. (2009) initiated the application of this technique to examination timetabling problems. A new model of granular examination to time-slot allocation was proposed using the pre-processed information from student-exam data for the capacitated and un-capacitated examination timetabling problem. The outcome of the pre-processed student-examination information was a set of conflict chains that could construct a simple scheduling task by assigning a group of examinations into a particular time-slot rather than one examination at one time. The scheduling cost was minimised by rearranging the examination spread matrix. The approach was tested on the University of Nottingham and the Toronto datasets and produced promising results.

2.2.3.5 Neural Networks

The idea of a neural network is inspired by the network of biological neurons that form an interconnected group of nodes, i.e. input, hidden and output layers (Haykin, 1999), information being processed between groups of nodes that are connected to each other based on a learning system. The effectiveness of the neural network in constructing an examination timetable was investigated by Corr et al. (2006). The approach applied graph colouring heuristics and the Kohonen self-organising neural network to train the regularities of the input data known as 'feature vectors'. Principally, the difficulty of each examination was measured using the neural network before proceeding to ordering and scheduling the most difficult examination first to a time-slot. The network divided the examinations into three categories of scheduling position - early, middle and late - during timetable construction. As the examinations were ordered, they were assigned to the time-slots adaptively. This approach has significantly created feasibility in the solution compared with a single graph colouring heuristic.

2.2.4 Meta-heuristic and Improvement Heuristic Techniques

Much research in the area of timetabling has utilised meta-heuristic approaches with great success. These techniques begin with one or more initial solutions and employ search strategies for the purpose of improving the solution quality (Petrovic and Burke, 2004). Essentially, various search strategies, for examples, tabu search, simulated annealing, genetic algorithms and ant colony optimisation, are designed to escape from local minima. Hybrid, meta-heuristic approaches have also been shown to be particularly effective. An overview of meta-heuristic approaches can be found in studies by Burke and Kendall (2005), Petrovic and Burke (2004) and Qu et al. (2009b). The meta-heuristic technique can be divided into two categories: local search-based techniques that deal with a single candidate solution at each iteration, and population-based that employ a population of candidate solutions during the search process. The next subsection discusses the categories of meta-heuristic approaches.

2.2.4.1 Local Search-based Methodologies

Local search approaches can search extensively in the search space and at the same time they often aim to avoid the search from being stuck in local optima. These algorithms can start the search with poor initial solution and the feasibility of the initial solution is not crucial as most of the computation time will be spent in the improvement phase. Carter et al. (1996) stated that this method drew a large potential in producing good quality results for timetabling but might have required a long running time for an optimal solution. These local search-based methodologies can be broken down into several approaches and are discussed in the following subsections.

Hill Climbing. The analogy of the hill climbing method illustrates the incremental changes of solution quality iteratively. Hill climbing is a type of local search which is very straightforward to implement, such simplicity making it popular for implementation in optimisation problems. It starts with the current solution in hand and generates neighbouring solutions, evaluating and replacing it with a candidate solution that is better. Nevertheless, this method can have a relatively poor performance due to being easily trapped in local optima. However, the hybridisation of hill climbing with other methods can be very successful, as demonstrated by Caramia et al. (2008), Merlot et al. (2003), Kendall and Mohd Hussin (2005a), Ross and Corne (1995) and Burke and Bykov (2008).

Caramia et al. (2008) proposed local-search based algorithms to solve capacitated and un-capacitated examination timetabling problems. Their study sought two main objectives: to minimise the time-slot number and to maximise the timetable quality for a fixed time-slot number. The algorithm started with a greedy scheduler assigning examinations to time-slots with respect to the conflict-free requirements. An examination with higher priority, i.e. high clashing, was selected first for an assignment. At this stage, the number of time-slots was increased to ensure that all of the examinations were scheduled into a timetable. Then, the hill climbing as a penalty-decreaser was used to improve the quality of timetable with the current number of time-slots or with decreasing number of time-slots until there was no further improvement. Hill climbing as the penalty-trader was used once again if there was no improvement in timetable quality by the penalty-decreaser. This approach was tested against the Toronto and Nottingham benchmark datasets and showed superiority by producing several best known results in the examination timetabling literature.

A hybrid algorithm for capacitated and un-capacitated problems of examination timetabling was presented by Merlot et al. (2003). The hybridisation consisted of three phases, namely, constraint programming, simulated annealing and hill climbing, the latter method being used to further improve the obtained solution. Their hybridisation technique was tested on the University of Melbourne problem and the Toronto and Nottingham benchmark datasets. In another study, Kendall and Mohd Hussin (2005b) demonstrated the implementation of a hyper-heuristic with hill climbing to solve examination timetabling problems.

Ross and Corne (1995) investigated the performance of a genetic algorithm compared with stochastic hill climbing and simulated annealing on different features of capacitated examination timetabling problems. They found that these approaches generated good solution quality in most problem features and were better than the genetic algorithm. Moreover, the study found that, while the genetic algorithm could generate a number of distinct solutions compared with stochastic hill climbing and simulated annealing because it was able to produce a pool of solutions at one time, the stochastic hill climbing and simulated annealing worked very effectively on difficult problems. They concluded that the stochastic method is generally suitable for investigating examination timetabling problems.

A new variant of hill climbing known as 'late acceptance strategy' was introduced by Burke and Bykov (2008) to the timetabling problem. The approach considered several objective function values in the previous steps as a reference in considering whether or not the current solution should be accepted. In contrast to the basic hill climbing, the late acceptance strategy listed several previous values with k size. The current solution had several choices for being accepted or rejected. The acceptance criteria followed the pure hill climbing where the current solution was accepted whenever it achieved better or equal performance. Once the current solution was accepted, it was included in the list and the last element of the list was discarded. The test on the Toronto benchmark datasets demonstrated that this approach was superior and able to produce several best results when compared with other approaches in the examination timetabling literature. The late acceptance strategy was then implemented within a hyper-heuristic framework by Özcan et al. (2009) with a combination of different heuristic selection methodologies. It was found that the combination of simple random heuristic selection with the late acceptance strategy could perform better than the other combinations.

Tabu Search. Tabu search was initiated by Glover (1986). This approach is based on the hill climbing algorithm in that it principally improves the current solutions by restricting certain moves using adaptive memories known as a 'tabu list'. Hence, this strategy could avoid the part of the procedure from cycling to the previous non-improving solution, as well as to explore more solution space. These tabu moves are forbidden for some occasions (usually a number of iterations) where the time resume is called 'tabu tenure'. Nevertheless, since there is a possibility that this approach could prohibit good moves, an 'aspiration criterion' is employed in order to override the tabu status of moves. The ideas of intensification and diversification are introduced in tabu search in order to search effectively for good solutions. A discussion on tabu search strategies was carried out in Glover and Laguna (1997), Glover (1989), Glover (1990), Pirlot (1996) and Gendreau and Potvin (2005).

Recently, the tabu search approach has been implemented in examination timetabling quite extensively and the few examples illustrated in this section have shown the success of this approach. White and Xie (2001) investigated the application of short term and longer term memory of tabu search to the examination timetabling problem using a four-phase system called OTTABU. It was found that the algorithm worked effectively when both types of memory were applied together and the use of longer term memory could improve the solution quality by 34% compared with short term memory alone. Furthermore, the active examinations were detected by the longer term memory based on the frequency of moves and the movement of these examinations was avoided so that cycling could be prevented and at the same time diversify the search. Typically, the active examinations could help to estimate the size of the tabu list. The idea of the relaxation of the tabu list was to find a new neighbourhood in order to seek a better solution during the search. Besides that, the four-phase algorithm also incorporated the diversification and intensification strategy in the search for a solution. The study was extended by White et al. (2004) who beside longer term memory, also considered a relaxation strategy to the tabu list. Whenever there was no improvement to the solution quality, the tabu list was emptied in order to force the search to the unvisited region. The approach was found to generate a better solution compared with the approach with short term memory or without relaxation strategy.

Di Gaspero and Schaerf (2001) examined the tabu search approach based on the feature of graph coloring problems in solving the examination timetabling problem. Additional information was obtained from the edges and the nodes of the graph with the adaptation of weight values that represented the number of students in conflict between two examinations and the number of students enrolled for an examination, respectively. The shifting penalty mechanism was incorporated in order to vary the weights of hard and soft constraint in the objective function so that the exploration of the search space was guided. This approach also integrated a dynamic neighbourhood selection based on a violation examinations list that infringed either only a hard constraint or hard and soft constraints. Moreover, the adaptation of the variable size feature of the tabu list showed that this approach, tested against two benchmark problems and random instances, has had promising results.

Di Gaspero (2002) extended the study presented by Di Gaspero and Schaerf (2001). A multi-neighbourhood algorithm was proposed to the tabu search approach that combined two types of moves i.e. 'recolor' and 'shake'. The aim of the different neighbourhood combination was to diversify the search so that it could escape from local minima. The proposed approach involved a three-phase algorithm. In the first phase, it was concerned with creating a feasible solution and at the same time optimising the objective function. At this stage, only examinations that contributed to the changes of cost incurred by violation of either hard or soft constraints were considered to be moved. This phase is known as 'recolor TS'. In the second phase, 'shake TS' aimed to create a new starting point by exchanging the time-slots of two whole groups of examinations at once. Finally, the 'kicker' was performed in order to find further improvement to the current solution. This approach obtained superior results compared with their previous approach and with the constructive approach by Carter et al. (1996).

The tabu search approach is hybridised with an exponential Monte-Carlo procedure by Sabar et al. (2009) where the study is an extension of work by Ayob and Kendall (2003). The tabu search was incorporated in order to limit the non-improving moves during the solution search. Moreover, a counter-strategy that counted the successive non-improving iterations was incorporated with the approach that determined whether the worse solution should be accepted or not. The approach was tested on the uncapacitated problem of the Toronto benchmark datasets and results demonstrating the success of the implementation with several best results were obtained.

Simulated Annealing. Simulated annealing is a popular local search approach that draws upon the concept of annealing material in metallurgy. The simulated annealing algorithm moves towards the global optimum by moving the current solution to a randomly selected neighbouring solution with certain probability. The probability is controlled by a temperature which decreases monotically with each successive iteration. In a simulated annealing algorithm, the process starts at a very high temperature to give a higher probability of acceptance to the solution quality, and as it cools slowly, the probability of rejecting worsening moves increases. Consequently, there are three main parameters involved in directing the search in order to improve the solution quality: initial temperature, cooling schedule and end temperature. This approach was proposed by Kirkpatrick et al. (1983) and further discussion on simulated annealing can be found in Aarts et al. (2005) and Pirlot (1996).

Simulated annealing has proved successful in the timetabling area. Studies by Thompson and Dowsland (1996b), Thompson and Dowsland (1996a) and Thompson and Dowsland (1998) investigated the utilisation of simulated annealing in solving the examination timetabling problem. Thompson and Dowsland (1998) examined the robustness of simulated annealing for the examination timetabling system, where the focus of the study was to apply different neighbourhood structures and cooling schedules to the basic framework implemented in their previous study (Thompson and Dowsland, 1996a,b). They also considered the effect of sampling bias to the obtained solution. The problem was divided into a two-phase approach. In the first phase, they attempted to find a feasible solution, while the second phase sought to optimise the soft constraints for a better solution quality. The approach was tested on various datasets from several universities with different real world features. Three different neighbourhood structures were tested: standard, Kempe-chain and s-chain.

The standard neighbourhood structure is a simple move of a randomly selected examination to a random feasible time-slot, and is the most common one used in solving timetabling problems (Abdullah et al., 2005, Burke et al., 2010a, McCollum et al., 2009). The Kempe-chain neighbourhood structure is determined by two subsets of examinations, i.e. H and H' in two different time-slots, t and t', where $t \neq t'$. Both of H and H' are not conflicting with each other and the time-slots t and t' are exchanged between the two subsets. The implementation of this neighbourhood structure is discussed in Chapter 6. On the other hand, the s-chain neighbourhood structure is a variant of basic Kempe-chain where the chain is more than two subsets considered in moving time-slots. The study considered the maximum chain, s is as 2, and found that the basic Kempe-chain neighbourhood outperformed other neighbourhood structures in terms of solution quality. It was concluded that the neighbourhood selection in a simulated annealing approach might lead to a better quality timetable.

Burke et al. (2004a) also discussed the application of simulated annealing and the great deluge algorithm in the case of time-predefined methods for examination timetabling problems. The main objective of their study was to incorporate user-defined parameters in the algorithm, i.e. the computational time and desired solution quality. Computational time and solution quality were the input parameters based on the users' preferences of the timetable outcomes. It was understood that the longer searches would have significantly improved the quality of solution when there was greater exploration of the search space by the algorithms. The time-predefined simulated annealing employed an additional time-predefinition algorithm in order to make sure that the approach did not converge too quickly. Further, the hybridisation of simulated annealing with other approaches in examination timetabling showed promising results. Examples of such applications are found in Merlot et al. (2003) and Duong and Lam (2004) and are discussed as constraint-based approaches.

Large Neighbourhood Search. Large neighbourhood search was originally proposed by Ahuja et al. (2001), when investigating solution search methodologies on a large neighbourhood structure in the area of combinatorial optimisation. Abdullah et al. (2007) were the first to implement this approach in solving the problem of examination timetabling. Their study examined the capacitated and un-capacitated examination timetabling problems. A number of disjoined cliques of examinations were formed using the partitioning method. A cyclic exchange operation was then implemented in order to design a large neighbourhood structure which moved examinations from one clique to another. In the next phase, an improvement graph was used to solve the examination timetabling problem by employing the modified shortest path label-correcting algorithm in order to identify the negative cost partition-disjoint cycles. The 'insert' and 'eject' moves were used in the algorithm. Furthermore, the exponential Monte-Carlo procedure (Kendall and Mohd Hussin, 2005a) was employed in the algorithm which accepts worst solution and non-improving moves with certain probability. This was to ensure that the algorithm was less likely to get stuck at local optima during the cycle. Nevertheless, some problems occurred during the cycling where improvement moves were kept in the tabu list for the capacitated examination timetabling problems which slowed down the algorithm. This occurred with datasets that had a large value of conflict density and it was suggested that a larger solution space was needed to solve difficult problems. The computational time for this approach is very expensive as the algorithm needs to explore a large solution space. As tested on capacitated and un-capacitated problems, the approach obtained a number of best results compared with other approaches applied to the Toronto benchmark problem.

Meyers and Orlin (2007) presented a survey on university timetabling problems which emphasized the very large-scale neighbourhood search (VLNS) techniques. The structure of a cyclic exchange neighbourhood and the way in which this algorithm works on certain problems was described. Three criteria of VLNS algorithm that had been used in several areas in the literature were identified, i.e. variable depth method, network flow-based methods and neighbourhood based on restriction. The success of the VLNS techniques was found to be based on the choice of good neighbourhood functions and the development of an effective heuristic method to search the neighbourhood for improving solutions. Furthermore, two new approaches that had a high potential in solving the timetabling problem were proposed. These new approaches, namely, optimised crossover in a genetic algorithm and functional annealing (combination of neighbourhood search method with simulated annealing) could be applied to the cyclic exchange operation in VLNS. Optimised crossover proved to be superior to several previous applications noted in the literature. The approach was operated by finding the best child from all possible sets of all children in the population. Hence, the best objective function value was considered in order to find the best child. Functional annealing was operated by joining the elements of neighbourhood search and the simulated annealing approaches, and incorporated the objective function to escape efficiently from local optima while the neighbourhood search heuristic provided a more effective search of feasible solutions. This algorithm has been successfully applied in VLSN and has demonstrated the performance in several application areas.

Variable Neighbourhood Search. Variable neighbourhood search (VNS) is a meta-heuristic technique inspired by Mladenović and Hansen (1997). It represented a systematic change of more than one neighbourhood structure during the search. In order to avoid being trapped in local minimum, VNS acted by jumping to a new neighbourhood from the current solution if a better improvement was found. More details on the VNS approach can be viewed in Chapter 6.

In their preliminary study, Mladenović and Hansen (1997) applied the VNS technique to the travelling salesman problem with and without backhauls in order to illustrate the success of the methodology. The problem description shows that the adaption of GENIUS heuristic with VNS technique can produce competitive results compared with the basic GENIUS. VNS requires several neighbourhood structures of a different nature. The neighbourhood structures are employed one at a time. If one fails to improve the current solution, the other one may still have a chance. The neighbourhood continues to be improved until there is no more improvement and this process will continue with other neighbourhoods. In a study by Hansen and Mladenović (2001), the order of neighbourhood structures is based on a pre-defined sequence.

Within the VNS approach, Hansen et al. (2001) introduced a decomposition strategy which was applied to the p-Median problem. Variable neighbourhood decomposition search (VNDS) divides the problem into a sequence of sub-problems which are generated from the different pre-selected sets of neighbourhoods. During the improvement phase, the neighbourhood will be changed only if there is an improvement on the sub-problem solution. Otherwise, the search continues from the incumbent in the first pre-selected neighbourhood. This technique obtained superior results when applied to large problem instances. It showed good results compared with the fast interchange approach that had also been implemented in the same study. It was concluded that this approach consumed less computational time than the basic VNS.

In the area of examination timetabling, Wong et al. (2005) implemented variable neighborhood descent using multiple neighborhood structures to the un-capacitated problem. Each neighborhood structure was integrated by a different local search operator in order to explore and exploit the search space of solutions. The aim was to balance the intensification and diversification during the search space exploration. This approach has been tested on several well known benchmark problems and has shown superior performance.

Recently, Burke et al. (2010a) demonstrated that VNS and a hybridisation with a genetic algorithm could produce a good quality solution and the best known results in the literature. In their study, the genetic algorithm imitated the concept of hyper-heuristic and case-based reasoning where this method did not directly apply to the problem but worked at the high-level of abstraction. Since the solution quality was dependent on the selection of the neighbourhood, the genetic algorithm worked intelligently by selecting the list of neighbourhoods from the VNS framework. Recent studies have concentrated on the implementation of VNS as high-level heuristic (Ahmadi et al., 2003, Qu and Burke, 2005), and are discussed in Section 2.2.5.

Great Deluge. The great deluge algorithm was first introduced by Dueck (1993). The algorithm belongs to a type of local search approach based on the analogy of the raising of a water level. This algorithm has some similarity with the simulated annealing approach in that it accepts worse solutions subject to certain requirements. However, this algorithm is much simpler than the simulated annealing since it requires fewer parameter settings i.e. the 'up' parameter that determines whether the solution will be accepted or not if its quality is less than or equal to the current solution of the minimisation problem. The algorithm starts with an initial water level - the initial solution quality based on the objective function value. During the process, the water level will decrease based on an amount of decrease ('up'). Dueck (1993) suggested that the best 'up' value should be small enough, i.e. less than 1% of decrease, so that the algorithm could spend more time in searching for good solution quality. A further review on the great deluge algorithm can be found in Dueck (1993).

Burke and Newall (2003) presented local search methods, namely, hill climbing, simulated annealing and the great deluge algorithm, to improve high quality initial solutions obtained from an adaptive approach (Burke and Newall, 2004) during the construction phase. In the study, several parameters on local search methods (initial ceiling values and initial average probabilities) were observed in order to identify their role in producing good solution when initiated with high quality solutions. Their study showed that the high quality initial solution influenced the final solution of the constructed examination timetable. The great deluge algorithm performed effectively when compared with the other two local search methods and it produced several good results on the Toronto benchmark datasets at that time. The study found that the parameter tuning was needed for both the great deluge and simulated annealing algorithm in order to enhance the performance of the constructed solutions.

Petrovic and Bykov (2003) presented a multi-objective technique for the examination timetabling problem. This technique drove the search through a predefined path in the criteria space in order to give direction to the solution search. A weighted sum cost function was incorporated within the great deluge algorithm and the weight of each criterion was changed dynamically during the search process. This approach produced the best results published in the examination timetabling literature at that time.

The great deluge algorithm was employed by Burke et al. (2004a) to enhance the acceptance criteria by accepting improving moves even if they were greater than some given levels implemented in hill climbing. This was to ensure that the algorithm explored further into the restricted region of the search space. The approach was investigated only on a single neighbourhood structure and it was shown to be very effective and superior regarding examination benchmark problems compared with other approaches.

Burke and Bykov (2006) extended their earlier study (Burke et al., 2004a), by introducing the flex-deluge algorithm for solving the examination timetabling problems. This approach was a modification of the great deluge algorithm and hill climbing where new acceptance criteria based on flexibility coefficient were introduced. The approach avoided certain moves adaptively in order to search for more solutions by flexibility change in the coefficient. In order to overcome the infeasibility problem, the approach employed Kempe-chains as used by Thompson and Dowsland (1996a). The approach was tested on the Toronto benchmark datasets and obtained several best results when compared with other best approaches in the literature.

In a recent study by Turabieh and Abdullah (2011), the great deluge algorithm was hybridised with a heuristic procedure known as the ‘electromagnetic-like mechanism’ within timetabling approaches. The proposed heuristic procedure was first introduced by Birbil and Fang (2003) and based itself on particle swam optimisation. It worked by forcing the search to a promising area by dynamically changing the decay rate of the great deluge algorithm utilising the calculation of the obtained solutions. The approach was very competitive where it obtained best results on the Toronto and ITC2007 benchmark datasets.

Greedy Randomised Adaptive Search Procedure. Greedy randomised adaptive search procedure (GRASP) is an iterative method that consists of two phases of algorithm, i.e. constructive and improvement phases, where at the end of iteration the best solution is returned. Compared with other constructive phases, GRASP involves two main processes - dynamic construction heuristic and randomisation (Blum and Roli, 2003). The solution construction is achived gradually where each element is chosen randomly from the top n in the list (called candidate list). The order of the remaining elements is based on its heuristic criterion.

An example of GRASP implementation in examination timetabling was provided by Casey and Thompson (2003) who developed an iterative method for solving the problem. The objective was to minimise the proximity cost of the timetable (Carter and Laporte, 1996). In the first phase, the greedy approach based on graph colouring heuristics introduced by Carter et al. (1996) was implemented. The next examination to be scheduled was chosen from the candidate list using roulette wheel selection and the chosen examination was assigned to the first available time-slot. The strategy was to launch the local search algorithm iteratively. If the examination could not be assigned to the time-slot due to a clash with other examinations, then the backtracking strategy was employed with the insertion of the tabu list in order to avoid cycling during the search. In the next phase, a limited form of simulated annealing was implemented using a drastic cooling procedure that started with a high temperature and cooled with a fast rate. In addition, the Kempe-chain based neighbourhoods (Thompson and Dowsland, 1996b) and memory function were incorporated. These improvement algorithms were used to maximise the quality of solution and create a diversification strategy in the search. The result showed that the initial solution generated by the saturation degree heuristic performed the best compared with other heuristics. The GRASP technique obtained competitive results among examination timetabling approaches and held one of the best results on the Toronto benchmark datasets in the literature.

The GRASP has also been applied within the hyper-heuristic framework by Burke et al. (2009). In their study, the examination timetable was constructed dynamically using a hybridisation of two graph colouring heuristics. The restricted candidate list was the list of examinations due to be scheduled next, created during the construction phase of GRASP. During the search, the examination from the list was chosen randomly. The size of the restricted candidate list was adaptively determined by the hyper-heuristic at each iteration. The study introduced a switching point of changing from the implementation of largest weighted degree to saturation degree adaptively. In the next phase, the solution was improved using steepest descent and was able to produce a good solution in a shorter time period compared with their tabu search approach.

Developmental Approach. The developmental approach, first introduced by Pillay and Banzhaf (2008) in solving un-capacitated problems of examination timetabling, is an approach that mimicks the cell biology process of growth and development of organisms. Their approach created a population of organisms starting from a single cell that referred to a time-slot in the timetable, and a number of processes related to cell division, interaction and migration were employed for constructing and improving the initial timetable. This was constructed based on saturation degree ordering which began with a single cell and examinations were assigned to feasible cells. On the other hand, the examinations with no feasible cell caused the cell to divide into two other cells which contained examinations that caused the clash and list of all examinations. In the next phase, the cell was migrated by swapping examinations between feasible cells and interacted with by exchanging the cell position in order to offer better or matured algorithm based on solution quality. The approach was tested on the Toronto benchmark datasets and obtained superior results when compared with other population-based approaches.

Pillay (2009b) continued to study the developmental approach proposed by Pillay and Banzhaf (2008) by improving the solution quality and also the processing time of the algorithm. The revised version included random elements to choose examinations in the cells instead of considering each pair of examinations and each examination during a cell migration and interaction process during the improvement phase. Furthermore, the ordering of examinations employed a new heuristic, namely, the highest cost introduced by Pillay and Banzhaf (2009), as a second heuristic to help in choosing the right examination in the list if there was a tie in saturation degree original ordering. The test on the un-capacitated problem of the Toronto benchmark datasets showed that the revised version of the developmental approach significantly improved the solution quality and reduced the processing time.

Harmony Search Algorithm. The harmony search algorithm, first introduced by Geem et al. (2001), used the idea of a musical improvisation process. Al-Betar et al. (2010) applied this approach to the un-capacitated examination timetabling problem; it had also previously been successfully implemented in the course timetabling problem (Al-Betar and Khader, 2009, Al-Betar et al., 2008). A new solution, known as 'new harmony', was obtained at each iteration and stored in 'harmony memory'. Meanwhile, in producing the solution, three main groups of criteria were considered: memory consideration, random consideration and pitch adjustment. Each of these criteria was associated with recombination, randomness and neighbourhood structure of the algorithm, respectively. Al-Betar et al. (2010) investigated the contribution of solution quality based on the various combinations of meta-heuristic components. Their study found that the combination of the three components produced the best results compared with other combinations of meta-heuristic components. When tested on the Toronto benchmark datasets, the performance of this approach was competitive compared with other approaches in the literature.

2.2.4.2 Population-based Search Methodologies

Recent research trends in timetabling have explored population-based search methodologies. The search within these methodologies is concerned with more than one different initial solution at any one time. A collection of initial solutions is called the 'population' and is treated simultaneously in generating a number of new distinct solutions. There are several population-based search methodologies within the examination timetabling problem: genetic algorithm, memetic algorithm and ant algorithm, each of which is discussed in the following subsections.

Genetic Algorithms. Genetic algorithms were inspired by the principles of evolution in nature that operate on the population of solutions to search problems (Glover and Kochenberher, 2003, Sastry et al., 2006). The solutions are represented by chromosomes where the solution quality is determined by the fitness function and random new generations of solutions are produced accordingly. The new generations of solutions are modified through reproduction progression by applying crossover (combining) or mutation (altering) operators iteratively in order to find a better solution. The definition of parameters is user-specific and each combination of parameters influences the performance of the approach.

A simple survey on genetic algorithms in educational timetabling was presented by Corne et al. (1994). The process of evolutionary algorithms within educational timetabling was briefly discussed with a focus on the two types of chromosome representation - direct and indirect. The survey raised several issues on the current implementation of chromosome representation, including their comparison and implementation in existing studies.

Since this approach generally promised successful implementations, it was employed in many studies of the examination timetabling problem. For instance, Burke et al. (1995) implemented a recombination operator of evolutionary algorithms with direct representation of a timetable and the property of feasibility was maintained during the search process. The graph colouring heuristic was hybridised with a crossover parameter in order to obtain a good timetable. In their study, new heuristics relating to genetic search space were designed in order to reduce the length of the timetable and to maximise the examination spread across the timetable period.

Ross et al. (1996) discovered a new phase transition relating to timetabling problems that focused on different homogeneity degrees and connectivity of graph colouring issues. The problem was tested with an evolutionary algorithm and stochastic hill climbing with the existence of phase transition in order to understand the performance of different algorithms. In another study, Ross et al. (1998) found that the genetic algorithm with direct encoding failed to work effectively on the examination timetabling problem due to a lack of success in exploring different areas. This conclusion led to the general method of searching for good algorithms rather than searching for a solution to a problem, with the introduction of a hyper-heuristic approach, discussed in Section 2.2.5.

The crossover operator in a genetic algorithm based on maximal clique in timetabling problems was investigated by Terashima-Marin et al. (1999). The aim of their study was to produce a better offspring from the mating parents and to search for a higher quality solution from the exploration of the selected search space. In order to create a larger solution space, two parts of the solutions from the clique were combined. The study was tested on twelve datasets, eleven of which were randomly generated with certain features, while the twelfth came from real data from the Department of Artificial Intelligence at the University of Edinburgh.

Erben (2001) applied a grouping strategy of genetic algorithms to the graph colouring and the examination timetabling problems. This algorithm implemented a tournament selection strategy where a set of examinations that were scheduled in the same time-slot was considered as a group. The mutation operator applied in the algorithm acted by swapping two groups of examinations chosen randomly in the chromosome, while other genetic operators worked for more than two groups or reversed the order of the scheduled groups. The study found that adequate representation of individuals in a genetic algorithm was very important, as was the fitness function that depicted the solution quality obtained. The results obtained with the standard benchmark examination timetabling problems showed that this algorithm obtained promising results.

A genetic algorithm was also used by Wong et al. (2002) and Sheibani (2002) to solve real world examination timetabling problems. For Wong et al. (2002), the problem was highly constrained and provided the motivation to map it into a constrained satisfaction problem combined with a genetic algorithm. On the other hand, Sheibani (2002) presented a genetic algorithm for solving the examination timetabling problem in a training centre. The aim of the study was to spread out the examinations within a schedule considering the relationship between examinations evaluated using an activity-on-arrow network. A standard genetic algorithm with crossover operator was employed where L length of two parents were randomly selected and mated. Both of the approaches were successfully implemented in the institutions.

In another study, Côté et al. (2005) investigated a hybrid bi-objective evolutionary algorithm with a local search operator for the un-capacitated examination timetabling problem. The timetables were generated, taking into account two objectives: 1) obtaining a good quality feasible timetable and 2) minimising timetable length. The evolutionary algorithm used in their study employed tabu search as a local search operator. A simplified version of variable neighbourhood descent was also applied to search for more neighbours of good solutions. At the early stage of the algorithm, the random initialisation was allowed to produce infeasible timetables. The tabu search algorithm was used to reduce the constraint violation from the pool of initial timetables produced during the early stage. The simplified version of variable neighbourhood descent was then performed in order to increase the solution quality using two neighbourhood structures, i.e. Kempe-chain and one move with tabu search. The Kempe-chain neighbourhood structure as explained by Thompson and Dowsland (1996a) acted as a chain of conflicting examinations and was swapped by interchanging two time-slots. This approach demonstrated that non-dominated timetables could be produced using different timetable lengths and it was shown to be superior for several tested benchmark problems.

Cheong et al. (2007) demonstrated a multi-objective evolutionary algorithm in relation to the examination timetabling of the capacitated problem. The approach was capable of tackling the problem that required minimisation of the timetable length while at the same time distributing the students in the timetable. A timetable could easily be generated without earlier information about the timetable length. A variable-length chromosome was introduced to change the length of examination session in a flexible way, while the day-exchange crossover was performed to diversify examinations in the time-slot of the day. The goal-based Pareto ranking was introduced in the study, which involved a two-phased process in order to choose the relative strength of solutions. The results showed that this approach could easily produce a feasible solution and was superior to several well-known approaches in the literature on benchmark datasets.

Memetic Algorithms. Within the population-based search methodologies, the memetic algorithm is a well-known approach. It is known as a hybridisation of evolutionary algorithm and local search methods (mainly hill climbing) which is used in order to improve the solution quality. The used of the memetic term was first initiated by Moscato in 1989 (Moscato and Cotta, 2003). There have since been several successful implementations of the memetic algorithm within the examination timetabling problems, including those by Burke et al. (1996b), Burke and Newall (1999), Özcan and Ersoy (2005) and Ersoy et al. (2007).

Burke et al. (1996b) combined an evolutionary algorithm with a local search method to study the capacitated problem of examination timetabling. The combination of light and heavy mutation was employed in the algorithm in order to reassign single and groups of examinations into a timetable. In the next phase, the hill climbing method was added after each of the mutation operators to improve the solution quality. The evaluation function in the problem penalised the unscheduled examinations heavily with the weight value of 2000, and considered the spread of the timetable by taking into account the number of conflicts between two time-slots on the same day. Although this technique showed great success, the computational expense increased due to the incorporation of the hill climbing method.

Burke and Newall (1999) implemented the memetic algorithm used in the previous study (Burke et al., 1996b) with a decomposition strategy for the examination timetabling problem. The purpose of this multi-stage approach was to split the large problems into a defined number of sub-components so that the algorithm could schedule the examinations all at once before proceeding to the next sub-component. Four large benchmark datasets were tested and it was demonstrated that the processing time could be reduced significantly while at the same time the solution quality was improved. Three graph colouring heuristics (colour degree, largest degree and saturation degree) were hybridised and the saturation degree showed good improvement among those heuristics. The backtracking and forward checking strategy were also incorporated if an infeasible timetable occurred in the early assignments.

A standard data format for the examination timetabling problem at the Faculty of Engineering and Architecture, Yeditepe University in Turkey, was proposed by Özcan and Ersoy (2005). In their paper, Final Examination Scheduler (FES) was introduced as a tool that accepted Timetabling Markup Language (TTML) as the input. The timetabling problem was solved using a memetic algorithm approach, introduced by Alkan and Özcan (2003), which employed a genetic algorithm that cooperated with modified violation directed hierarchical hill climbing (VDHC). The VDHC strategy worked by creating a hill climbing approach for each type of constraint. The entire set of hill climbing approaches were then gathered under a single hill climbing approach named VDHC. Three hierarchical levels of resolution were introduced to change the resolution level depending on the fitness function.

Ersoy et al. (2007) proposed a hyper-heuristic methodology known as 'hyperhill-climber' that adapted within a memetic algorithm for selecting the best hill climber in examination timetabling problems. The hill climber determined the best ordering for the successive application of hill-climbers. In their study, three types of hill climbers were introduced, i.e. deterministic, adaptive and self-adaptive, which were evaluated within a memetic algorithm in different orderings. The results were tested on the six problems of the Toronto benchmark datasets and the approach obtained promising results. The study found that the deterministic hyperhill-climber with the implementation of a single hill climber at any one time performed the best. This approach is an extension of their previous work in studies by Alkan and Özcan (2003) and Özcan and Ersoy (2005).

Ant Algorithms. The idea of the ant algorithm was inspired by the concept of real ants foraging for food, and was first introduced by Marco Dorigo (Dorigo and Blum, 2005, Dorigo et al., 1996). The ant moves to find food and produces pheromone trails when searching for the shortest way from the food sources to their nest. The path with a higher pheromone level gives an indication of the quality and quantity of the food brought back by the ants from the sources. This has obviously guided the other ants in their search for more food from the new discovery of food sources.

The ant algorithm has been used to solve hard combinatorial optimisation problems including timetabling, and surveys on the application of the ant algorithm can be found in Dorigo and Blum (2005). A study by Eley (2007) which employed an ant algorithm in solving the examination timetabling problem, incorporated the ant systems and the max-min ant systems with two randomised strategies in order to find the constructive heuristic and the pheromone trail. During the test, different parameter settings, i.e. the number of cycles, the weighting factors and the evaporation rate, were required in order to make sure that the algorithm worked effectively. The approach was tested on the Toronto benchmark datasets and obtained encouraging results.

This approach is also studied by Azimi (2004) and Azimi (2005). In the earlier study, ten randomly generated datasets of examination timetabling problems were investigated using simulated annealing, tabu search, genetic algorithms and the ant colony system within a similar framework. The initial solution for each algorithm was created randomly, while the ant colony system was created based on heuristic figures. A comparison of these approaches showed the ability of the ant colony system to produce a good quality solution when demonstrated under a given running time. In this study, the tabu search was found to be more effective than the ant colony system in improving the solution. The hybridisation of the ant colony system was further examined in his later study (Azimi, 2005). Comparisons of three different hybridisations of the ant colony system and the tabu search with other single local search algorithms demonstrated the superiority of the approach. The study found that the sequential ant colony system and tabu search was the best variant for the tested problem. During the search, the effect of pheromones from the ant colony system caused the solution to converge. Tabu search was therefore applied in order to disrupt the search. Nevertheless, the success of this approach could not be compared with other approaches in the literature since it was not tested against benchmark datasets.

The ant algorithm was also used by Dowsland and Thompson (2005) to solve the examination timetabling problem within graph colouring theory. The aim of their study was to find the minimum number of time-slots for the problem by considering the clash free requirement as the hard constraint. The proposed variants of ant algorithm were based on two graph colouring heuristics - recursive largest first and saturation degree. Different trail calculations were also considered, including one that was introduced for the purpose of diversification in the search and to avoid the occurrence of infeasible timetables. Moreover, two statistical measurements were used to evaluate the performance of the introduced combinations and the approach was comparable to others in the literature. The study suggested that the incorporation of time-windows, seating capacity and second-order conflicts could be considered during the investigation.

2.2.5 Hyper-heuristic and Case-based Reasoning Techniques

One disadvantage of the approaches described in the previous section is that there is often a reliance on parameter tuning in the production of solutions in particular circumstances. Moreover, most of the approaches are problem dependent and need a different setting of parameters, revealing the inconsistency across various instances. This drawback has motivated the introduction of more general methodologies i.e. hyper-heuristic (Burke et al., 2003c, Ross, 2005) and case-based reasoning (CBR) (Burke and Petrovic, 2002, Petrovic and Burke, 2004) and they are discussed in the following subsections.

2.2.5.1 Hyper-Heuristic

A hyper-heuristic is a search methodology that has received recent attention for timetabling. One of the motivating goals is that it should be a more general approach than other meta-heuristic approaches in the literature. Burke et al. (2003c) defined a hyper-heuristic as "the process of using (meta-) heuristics to choose (meta-) heuristics to solve the problem in hand". Recently, Burke et al. (2010b) presented a more general definition - "A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems" - that reflects current research trends. Generally, the hyper-heuristic works as a high-level heuristic and intelligently chooses a set of low-level heuristics based on learning mechanisms. The low-level heuristics are usually a set of simple heuristics that have different potential (either strength and weaknesses) on different problem instances. The idea of using sets of low level-heuristics allows the approach to work on various characteristic of problems.

Throughout the search process, some learning mechanisms are embedded in the given information for identifying the best low-level heuristic to be used. The learning mechanism could be on-line or off-line or there could be no-learning at all (Burke et al., 2010b). Essentially, the on-line learning happened to have an effect during the search process while the off-line learning involved an initial learning mechanism on the training problem. Thus, the generated information from the off-line learning could be used to solve the problem in hand. The no-learning mechanism involved no learning to the problem and so no information is needed to solve the problem. Generally, the hyper-heuristic approach can be categorised into two methodologies, i.e. heuristic selection and heuristic generation. This classification is based upon the nature of the heuristic search space. An overview of the hyper-heuristic approach can be found in Burke et al. (2003c), Petrovic and Burke (2004), Ross (2005) and Burke and Kendall (2005). The following subsection discusses the classification of hyper-heuristic approaches.

Heuristic selection methodologies. Heuristic selection methodologies involve a selection of existing heuristics in a hyper-heuristic framework. Usually, the hyper-heuristic aims to choose the most appropriate problem-specific constructive heuristic that already exists in the framework, and continue to search for a solution with a perturbation heuristic. Within hyper-heuristic approaches, the most common constructive heuristics used in examination timetabling are graph colouring heuristics and moving strategies. The successful application of hyper-heuristics as the heuristic selection methodologies has increased recently in studies of examination timetabling problems. Within these approaches, there are several heuristic selection methodologies reported in the literature - for example, simple random, random descent, greedy search, choice function, reinforcement learning, tabu search and the Monte-Carlo procedure.

The tabu search hyper-heuristic selection employs a certain length of tabu list of low-level heuristics to prevent the non-performed low-level heuristic being chosen too quickly and to ensure that other potential heuristics could be applied. Examples of the tabu search hyper-heuristic selection in examination timetabling are provided by Kendall and Mohd Hussin (2005a), Burke et al. (2005a) and Burke et al. (2007). In the first of these studies, Kendall and Mohd Hussin (2005a) and Kendall and Mohd Hussin (2005b) investigated the implementation of the tabu search hyper-heuristic framework for examination timetabling problems at the University Technology MARA (UiTM), Malaysia and Toronto benchmark datasets. The approach considered several graph coloring heuristics and moving strategies (i.e. 1-opt or 2-opt) as the low-level heuristics. Essentially, the low-level heuristics acted as a strategy to allow movement through a solution space. A number of low-level heuristics was stored in a fixed length of tabu list whenever they could not improve the solution quality. The approach was tested on eight problems of the Toronto benchmark and obtained good quality solutions across the problem range. In real world implementations, their solutions were compared with the manually generated timetable, indicating that their approach was better in at least 80% of cases in terms of solution quality.

Burke et al. (2005b) proposed hybrid graph colouring heuristics within a hyper-heuristic framework for constructing examination timetables. Two graph colouring heuristics - largest degree and saturation degree - were employed as low-level heuristics. In their study, the tabu search approach was used as a perturbation heuristic within hyper-heuristic. These graph colouring heuristics were intelligently selected for constructing solutions. Within the tabu search hyper-heuristic, a knowledge based approach, namely, case-based reasoning, was hybridised to choose graph colouring heuristics based on the knowledge of the appropriate heuristic. The approach was tested on random and Toronto benchmark datasets and the results were comparable to other best approaches. Furthermore, Burke et al. (2007) investigated a multi-stage hyper-heuristic where the graph colouring heuristics had been permutated into two stages. Graph colouring heuristics as low-level heuristics were applied in order to obtain solutions based on a learning mechanism and the solutions then were modified by high-level heuristic indirectly. Tabu search worked as a high-level heuristic which indirectly modified the solutions in hand rather than operating directly on the problem. Several permutations of graph heuristics were employed to construct solutions for examination and course timetabling. The study found that the increasing number of low-level heuristics would significantly increase the solution quality. However, the computational time would also increase due to the growing size of the search space. The approach produced competitive results on both examination and course benchmark timetabling problems.

The variable neighbourhood search can also be used as a high-level heuristic within the hyper-heuristic approach. A study by Ahmadi et al. (2003) described the application of variable neighbourhood search as a perturbation-based algorithm to the examination timetabling problem. In order to construct solutions, the study incorporated weight values to low-level heuristics, namely, examination selections, time-slot selections and room selection. Later, Qu and Burke (2005) investigated the application of a hybrid variable neighbourhood search to choose the sequence of low-level heuristics, i.e. graph colouring heuristics for examination timetabling problems. The two neighbourhood structures used consisted of a graph colouring heuristic sequence that changed to single or double flipped. The approach was tested on the Toronto benchmark datasets and its performance compared with different perturbation heuristics: tabu search, steepest descent method and iterated local search. Of these, the iterated local search performed the best compared with other approaches. The study found that even though a good perturbation heuristic was used, it may not necessarily found good quality solutions. It was concluded that good quality solutions could be obtained when a huge search space was used.

Other studies within the hyper-heuristic approach employed a genetic algorithm as the heuristic selection methodology. A study by Ross et al. (2004) used a genetic algorithm to search iteratively for the nearest labelled point within a simplified search space of problem-state-description. Each label in the algorithm described each heuristic in the list, where two different categories of heuristics were used - event-picking and slot-picking. In order to evaluate the overall performance of each heuristic, three different fitness measures were introduced on violation scores considering the pairs of event-picking and slot-picking heuristic. The algorithm was demonstrated on examination and course timetabling problems and showed promising results. The approach was judged to be very fast in solving the given problems.

Several heuristic selection methodologies, acceptance criteria and their combinations within the timetabling problem were investigated by Bilgin et al. (2007). The study considered simple random, random descent, tabu search, choice function and greedy as their heuristic selection methodologies, and the performance of thirty-five combinations of different mechanisms of the hyper-heuristic were compared in depth. The result showed that the combination of heuristic selection and a move acceptance strategy did have an impact on the solution quality. Further, the combinations were found to work differently with diverse problem instances. The analysis indicated that some of the combinations worked better on different objective functions.

A reinforcement learning method presented in a study by Nareyek (2003) has been used by Özcan et al. (2010) to choose the low-level heuristics based on a reward and punishment scheme known as 'utility value'. This learning mechanism employed an adaptation scheme that was based on the move acceptance by a remembrance mechanism whether to remember or to forget. The forgetting mechanism creates the upper and lower bound of utility value. This strategy was combined with the great deluge algorithm as a move acceptance criterion. The Monte-Carlo hyper-heuristic was investigated by Burke et al. (2010c) in relation to un-capacitated problems of the Toronto benchmark. The Monte-Carlo procedure was incorporated as a learning mechanism to select heuristic components. The approach also incorporated simulated annealing with reheating mechanism as the move acceptance strategy.

The heuristic selection methodologies could be based on the performance of low-level heuristics . Qu et al. (2009a) introduced an adaptive heuristic hybridisation for constructing examination timetables and solving the graph colouring problem. The approach was named 'random iterative graph-based hyper-heuristic' and used to construct various solution qualities using different heuristic sequences. The heuristic sequences consisted of several graph colouring heuristics: saturation degree, largest weighted degree, largest enrolment and largest degree. During the solution construction, a sequence of heuristics was developed randomly and the sequence was analysed in terms of feasibility. The infeasible sequence was put aside and only the feasible sequence was used for further analysis. The study observed that the hybridisation of the largest weighted degree and the saturation degree heuristics could produce good solution quality especially at the early stage of solution construction. The study also analysed the way in which graph colouring heuristics were automatically hybridised. Comparison with other approaches to examination timetabling and graph colouring problems has shown that the hybrid heuristics, though general, are very competitive.

Meanwhile, Pillay and Banzhaf (2009) presented a hierarchical concept of the hyper-heuristic where several heuristics were combined and applied simultaneously. The combination of heuristics was divided into primary and secondary heuristics. The primary heuristics included several graph colouring heuristics with one of these being identified as a priority heuristic, i.e. an important entity while performing the Pareto combination among the heuristics. When executing the comparison, the second heuristic was used in the timetabling process. This approach was tested on the Toronto benchmark datasets and produced several best results when compared with other hyper-heuristic approaches. Furthermore, the same study introduced a new heuristic known as 'highest cost' which was defined dynamically during the timetabling process. Based on this new heuristic, the most difficult examination was identified by obtaining the examination cost evaluation function introduced by Carter et al. (1996) and the examination with the highest cost value was scheduled first in the timetable. This heuristic was used as one of the low-level heuristics employed in the hyper-heuristic implementation.

Heuristic generation methodologies. So far, few studies have utilised heuristic generation methodologies in order to solve timetabling problems. An example of such an approach is demonstrated by Pillay and Banzhaf (2007) for generating low-level heuristics using genetic programming for the un-capacitated examination timetabling problem. A number of low-level heuristic sequences were evolved using the genetic operator whereby the new sequence of low-level heuristics was used to construct examination timetables. The approach was tested on the Toronto benchmark datasets and was able to produce a feasible timetable for all tested problems. An extension of this study by Pillay (2009a) employed genetic programming to evolve a function for generating a combination of low-level heuristics based on the scheduling difficulty. This combination was used hierarchically in constructing examination timetables. Both studies showed that the generation of a hyper-heuristic could be successfully applied to the timetabling problem.

Pillay (2010a) continued to investigate the evolving hyper-heuristic. The study presented a highly constrained examination timetabling problem of the ITC2007 which implemented this approach. It found that the evolutionary algorithm combined with three chromosome representations simultaneously performed better than the single chromosome representation in evolving low-level heuristics. A number of low-level heuristics were used to evolve a new combination of heuristics. The results demonstrated the success of the approach when compared with other approaches within the ITC2007. However, the study did not take into account the restrictions on the running time.

2.2.5.2 Case-based Reasoning

Case-based reasoning (CBR) is a knowledge-based system that collects information from previous problems and stores the solutions in the case base. In solving current problems, the most similar cases are retrieved from the case base using a similarity measure. An overview of CBR on educational timetabling can be found in Burke and Petrovic (2002) and Petrovic and Burke (2004).

In their application of CBR approaches to timetabling problems, Petrovic and Burke (2004) employed the previous knowledge by presenting it in terms of two important roles, i.e. solution reuse and methodology reuse. In a further study, CBR was investigated as a heuristics selection for the examination and course timetabling problems (Burke et al., 2006) where several well-known heuristics that worked effectively on timetabling problems were stored in the case base and used to solve the current problems. The low-level heuristics used were largest degree first, largest degree with tournament selection, colour degree, saturation degree and a hill climbing heuristic. The CBR system was constructed by two stages approach. In the first stage, problem learning using specific heuristics was involved and the best heuristics found in the case base were used for the current problem. In the second stage, the source case was revised and the unwanted heuristic was removed in order to avoid any confusion during the retrieval process. The study demonstrated that CBR could work at the level of generality in solving timetabling problems by choosing the right heuristic to be employed.

The CBR approach was also implemented by Yang and Petrovic (2004) as a method to decide a suitable graph colouring heuristic for initialisation strategy before proceeding to the next phase that employed great deluge algorithm. Their study proposed a similarity measure of two different graph representations based on fuzzy set (Zadeh, 1965). The approach proved to be successful where it obtained several best results when tested on the Toronto benchmark datasets. The similarity measure was further discussed in Burke et al. (2004b) for implementation in the CBR system, when it was used to choose the problem-solving method based on the features of timetabling problems.

2.2.6 Multi-criteria and Multi-objective Techniques

In the area of timetabling, the perception of educational timetabling as a multi-objective problem is not common even though in real world application many criteria arise in solving the problem. The approach considers several criteria for making decisions and is able to handle these simultaneously. Many approaches combine multiple objectives into a simple linear combination by using weighted aggregating functions in order to obtain a solution. However, it is often preferable to present several compromise solutions to the decision-maker in order to fulfill the various requirements (Landa-Silva et al., 2004). To perform this strategy, an intention is given to the Pareto optimisation technique as a method for multi-objective problems. It tries to find a set of compromise solutions that represent a good approximation to the Pareto optimal front. The Pareto-based meta-heuristic was introduced in order to overcome the difficulties in establishing the preferences among the criteria before the search. The evolutionary algorithm was reported as a method for the Pareto optimisation technique in multi-objective timetabling problems.

The survey paper by Landa-Silva et al. (2004) discussed a multi-objective meta-heuristic approach with Pareto optimisation for scheduling problems, i.e. machine scheduling, educational timetabling and personnel scheduling. The Pareto optimisation technique was preferable for producing various compromise solutions with a good approximation to the Pareto optimal front for the scheduling problem. Recently, many researches have used this technique to solve the multi-objective problem.

Recent studies in examination timetabling attempt to engage with the multi-criteria decision-making technique especially in solving real world problems. This technique has shown a significant achievement in the timetabling arena. Burke et al. (2001) applied a multi-criteria approach to examination timetabling by splitting nine criteria into three categories relating to: 1) room capacities, 2) proximity of examinations and 3) time and order of examinations. This approach was based on compromise programming (Zeleny, 1974) with regard to the concept of an ideal point defined in the criteria space and the mapping into the preference space. Consequently, the approach accepted constraints by taking into account the relative importance expressed by the timetable officer. The distances from the approximate ideal point indicated the quality of timetable that was obtained after considering all criteria. A hybrid of heavy mutation and hill-climbing was used as the search algorithm. The study was continued by Petrovic and Bykov (2003) who combined it with the great deluge algorithm for solving examination timetabling. This technique considered many criteria and improved the reference solution by changing the weight of criteria dynamically in order to direct the search. During the process, the search was driven through a predefined trajectory in the criteria space to approach the solution to the origin (ideal point) as possible. The variable weights great deluge algorithm produced some superior results to those that were published in the literature at that time.

An evolutionary algorithm within the multi-objective of real world problems has shown high rates of success. Paquete and Fortseca (2001) investigated the application of the multi-objective evolutionary algorithm to examination timetabling problems at the Unit of Exact and Human Science, University of Algarve. The approach used direct representation and Pareto ranking of population to evaluate each objective of the problem. Further, Côté et al. (2005) investigated a hybrid bi-objective evolutionary algorithm with a local search operator for the un-capacitated examination timetabling problem. The timetables that were generated considered two objective functions, namely, creating a clash-free requirement while satisfying student spread and minimising timetable length. In another study, Cheong et al. (2007) demonstrated a multi-objective evolutionary algorithm in relation to examination timetabling for capacitated problems. The approach was capable of tackling the problem by minimising the timetable period and at the same time distributing the students in the timetable.

A recent study by Asmuni et al. (2007) proposed a new fuzzy evaluation function for examination timetabling which involved multiple decision criteria. In addition to the most common objective used in examination timetabling i.e. the average penalty per student, the highest penalty imposed on any of the students was also considered to evaluate the quality of timetable solutions. They showed that fuzzy reasoning could be successfully applied to multiple decision criteria problems. The properties of multi-objectives for a new examination track for the ITC2007 was investigated by Burke et al. (2008). In order to overcome the complexity of multi-dimensional problems, the seven soft constraints introduced in the objective function were separated into two categories: (1) student and (2) administration. The trade-off between student and administration preference was investigated using the integer programming solver CPLEX 10 so that the Pareto-optimal solutions could be found. From the result, it was observed that there was a standard trade-off between the student and administrative preferences. More information on multi-criteria decision-making approaches in educational timetabling is cited in Burke and Petrovic (2002), Petrovic and Burke (2004) and Landa-Silva et al. (2004).

2.3 A Survey of Research on the Second International Timetabling Competition (ITC2007)

The International Timetabling Competition was first introduced in 2002 and its success led to a second competition in 2007 where three different tracks of problems were introduced: examination timetabling, post-enrolment-based course timetabling and curriculum-based course timetabling (McCollum et al., 2008). These problems described the real world implementation of timetabling with more problem constraints taken into consideration. The problem required time restriction while generating timetables as stated in the competition rules.

Since the introduction of the ITC2007 datasets, the problems have been actively investigated. This survey focuses solely on the examination timetabling track and the aim is to give an overview of the approaches to the datasets that were introduced. More details on the problem descriptions and constraints are provided by McCollum et al. (2008) and McCollum et al. (2010). The characteristics and constraints examination timetabling track are also discussed in Section 2.5.2. In this section, the implementation by the five winners of the competition (Atsuta et al., 2008, De Smet, 2008, Gogos et al., 2008, Müller, 2009, Pillay, 2008), together with the current implementations of the introduced datasets, are briefly discussed.

The first winner of the competition, Müller (2009) used the hybridisation of great deluge with other local search with great success for solving the three problems of ITC2007. The objective of the study was to build a general framework of algorithms that worked on various problem instances. A feasible solution was constructed employing an iterative forward search together with conflict-based statistics (Müller et al., 2004) in order to avoid revisiting the same solution. Hill climbing was then used to find the local optimum until no further improvement was observed. The solution continued to be enhanced by the great deluge algorithm where, during implementation, the bound was decreased using the cooling rate. Once the bound had reached the defined lower limit, the simulated annealing algorithm was employed to improve the solution quality. The process continued to work by repeating the hill climbing algorithm until it reached the time allocated. The results demonstrated the successful implementation of the approach and were shown to be superior in the three tracks of the ITC2007.

Gogos et al. (2008), the second winners of the ITC2007, presented a multi-stage approach. In the first phase, the approach was able to produce good quality feasible solutions during the solution construction. GRASP, a two-phase approach for constructing and improving solution quality, was applied. The solution construction employed five graph colouring heuristics (Burke and Petrovic, 2002) and the examination ordering was based on adaptive ordering introduced by Burke and Newall (2004). A backtracking procedure was incorporated during the construction phase whenever no feasible assignment was found. Once the feasible initial solution was obtained, the simulated annealing with Kempe-chain (Thompson and Dowsland, 1996a) and a reheating procedure began to improve the solution quality. Tabu search was incorporated during the search process to avoid cycling moves. Moreover, the solution was further improved with an integer programming approach using an open source mathematical solver GLPK based on the branch and bound procedure; it obtained superior results in the competition.

In the third place, Atsuta et al. (2008) proposed a general purpose solver for solving the three tracks of the ITC2007. The approach employed a hybridisation of tabu search and iterated local search combined with a constraint satisfaction solver. The hybrid algorithm incorporated a dynamic weighting scheme in order to improve the solution quality and the approach has been proven to work effectively on the ITC2007 datasets. This general purpose solver has also been explained in studies by Ibaraki (2008) and Ibaraki (2010) and has been successfully implemented in other timetabling problems.

The tabu search algorithm with an open source framework known as 'drools-solver' was applied by De Smet (2008) to generate a solution for the examination track of ITC2007. Three types of neighbourhood structure were employed, consisting of time-slot move, room move and examination switch. Examinations that had already been visited were kept tabu for a certain time. This approach was ranked fourth in the competition.

The fifth placed competitor was Pillay (2008) who proposed a developmental approach that mimicked the growth and development of organisms in cell biology. The approach involved a number of processes related to cell division, interaction and migration for constructing and improving the obtained timetable. As the process started, examinations were ordered with a saturation degree heuristic and each of the examinations was assigned to the feasible and lowest cost of cell (i.e. time-slot) and in the cases where more than one good cell was available, the cell was chosen randomly. From the best cell, the room was chosen based on the best fit heuristic. Meanwhile, the phase of cell division started whenever an examination could not be assigned into a time-slot. During the division phase, the cell was divided into two and the violated examination was set to be in a new cell. The cell was rearranged in order to lower the timetabling costs. If the constructed timetable was infeasible, then the cell interaction phase started in order to produce a feasible timetable. Finally, the timetable was improved in the migration phase by swapping two cells of the same duration. This approach has also been discussed in Pillay and Banzhaf (2008) and Pillay (2009b) in Section 2.2.4.2.

In the current implementation of the post-competition of the ITC2007, McCollum et al. (2009) successfully implemented an extended great deluge approach to the examination timetabling track. It was a two-phase approach where the first phase was concerned with the creation of a feasible initial solution using the adaptive strategy introduced by Burke and Newall (2004). In the second phase, the extended great deluge approach was applied using a move and swap heuristic while maintaining feasibility. A reheat mechanism of the great deluge algorithm was employed whenever there was no improvement in the solution quality. In the study, the boundary ceiling was set to be greater than the current best value and the decay rate was set to be higher in order to search for further better solutions. This study showed that the ITC2007 datasets were able to produce a feasible solution for all problem instances and the approach obtained six best results out of the twelve instances when compared with other ITC2007 approaches at that time.

A study by Burke et al. (2010f) improved the constructed timetable using the hyper-heuristic framework based on the idea proposed by Qu et al. (2009a). Examinations that contributed to the penalty cost of a timetable were identified and ordered using the hybridisation of graph colouring heuristics. The study found that the saturation degree with the largest weighted degree as a tiebreaker generated the best sequence while hybridisation of Kempe-chain moves with swapping time-slots led to better improvement to the solution quality. The Toronto benchmark datasets and the examination track of ITC2007 were tested to show the generality of the proposed approach.

Gogos et al. (2010a) further improved the approach after their first implementation in the competition by incorporating a few stages to improve the solution quality. Before the search continued with simulated annealing, a hill climbing approach was incorporated with Kempe-chain neighbourhood. Additionally, the improved approach employed a shaking procedure if there was no improvement to the current solution for a certain time, passing back to a simulated annealing phase. Within the time limit, the approach was able to produce competitive results and obtained four best results out of eight problem instances, with the exception of the hidden datasets.

In an earlier study, Gogos et al. (2009) had investigated the grid resources approach for solving the examination track of the ITC2007. This had distributed the problem instances to be solved into a number of nodes and had attempted to solve the problem simultaneously using a different temperature parameter for simulated annealing. The later approach, however, used a simple swarm-inspired logic that continuously improved the solution. The result demonstrated that the continuous improvement approach performed better than the first approach and the results were comparable to the best results in the literature. Gogos et al. (2010b) similarly implemented the distributed approach in considering the same problem, but this time the scatter search, i.e. a population-based approach, was used to solve the problem. The scatter search was hybridised with a path relinking strategy that connected two timetables, transforming them to an improved timetable where one of the timetables acted as a reference. Other approaches that were also tested with the examination track of ITC2007 include Burke et al. (2010f), Pillay (2010a) and Turabieh and Abdullah (2011) and have been discussed in the previous section.

2.4 Real World Examination Timetabling Datasets

Several real world datasets have been widely introduced to the examination timetabling community with variants of measurement and more practical constraints that represent the real world problems. These datasets were tested on a variety of approaches and a comparison was made for the purpose of scientific research. Table 2.2 shows the list of datasets from universities that were introduced in the literature.

Table 2.2: The examination timetabling datasets from different universities

Pioneer	University
Carter et al. (1996)	Carleton University, Ottawa; Earl Haig Collegiate Institute, Toronto; Ecole des Hautes Etudes Commercials, Montreal; King Fahd University, Dharan; London School of Economics; Ryeson University, Toronto; St. Andrew's Junior High School, Toronto; Trent University, Peterborough, Ontario; Faculty of Arts and Sciences, University of Toronto; Faculty of Engineering, University of Toronto; York Mills Collegiate Institute, Toronto
Burke et al. (1996b)	University of Nottingham
Ergül (1996)	Middle East Technical University
Wong et al. (2002)	École de Technologie Supérieure
Merlot et al. (2003)	University of Melbourne
Kendall and Mohd Hussin (2005b)	University of Technology MARA
Özcan and Ersoy (2005)	Yeditepe University
Kahar and Kendall (2010)	Universiti Malaysia Pahang

The first problem was introduced by Carter et al. (1996) with thirteen sets of problems from various universities around the world. These benchmark datasets were used widely as test beds in the examination timetabling community, introducing different problem dimensions and characteristics. These datasets are publicly available and can be accessed at `ftp://ftp.mie.utoronto.ca/pub/carter/testprob/`. A significant contribution by Carter et al. (1996) was the introduction of a penalty cost proximity function to evaluate the quality of examination timetable. The penalty cost was imposed by the number of students distributed across the timetable. A minimum number of time-slots for each benchmark dataset was introduced to this dataset for the purpose of solution quality assessment. Since there was a problem relating to the circulation of datasets under the same name, Qu et al. (2009b) introduced notations to differentiate various versions of the datasets. In the present thesis, the notation introduced is adapted to specify the datasets and version I is used as a test bed to the proposed approaches. For further details on the datasets and the problem description see Section 2.5.

The University of Nottingham benchmark dataset was first introduced by Burke et al. (1996b), who, in a further study (Burke et al., 1998b) added more constraints that considered consecutive examinations overnight. The original problem involved room requirements and capacities and at the same time took into account the minimisation of students attending two consecutive examinations per day. This dataset can be reached at `http://www.asap.cs.nott.ac.uk/resources/data.shtml`. An earlier study by Burke et al. (1993) had incorporated real world features: eliminating students sitting two consecutive examination periods, minimising disturbance during examination session, assigning the examination to a special room facility and employing variable size of time-slots so that examination length could be reduced.

Ergül (1996) initiated the datasets which involved two real world instances of the examination timetabling problem at the Middle East Technical University. Firstly, there were 682 examinations, while the second instance concerned a larger problem with 1449 examinations and with more constraints. However, some of these constraints were ignored since the implemented system did not take them into account. Study by Wong et al. (2002) introduced the examination timetabling dataset from the École de Technologie Supérieure for four departments of the engineering school in Montreal. In another study, Kendall and Mohd Hussin (2005b) drew attention to the examination timetabling problem for the Universiti Technology MARA in Malaysia. This dataset is unique since the university has more than one hundred campuses all over the country and an extensive timetabling task was involved. An additional constraint posed by the state public holiday was also introduced to the dataset.

Two datasets from the University of Melbourne which related to semesters 1 and 2 for the year 2001 were proposed by Merlot et al. (2003). Two new additional hard constraints were introduced, i.e. the availability of an examination and scheduling large examinations first. The objective was to produce a feasible good quality timetable. The examinations were required to schedule two time-slots per day for five working days and also to fulfill the room capacity requirement for each examination session. The datasets are available at `http://www.or.ms.unimelb.edu.au/timetabling`.

Özcan and Ersoy (2005) presented examination timetabling datasets from the Faculty of Engineering and Architecture at Yeditepe University. Two sets of problem instances for two educational years from 2001 until 2003 were used for solving the timetabling problem, namely, the requirement that students were distributed over the schedule and the room capacity constraint. The objective was to minimise the students sitting two consecutive exams on the same day. The datasets can be accessed at `http://cse.yeditepe.edu.tr/~eozcan/research/TTML`.

The most recent benchmark dataset of educational timetabling, presented by McCollum et al. (2008) and McCollum et al. (2010) originates from the Second International Timetabling Competition (ITC2007), which was the continuation of the previous competition first introduced in 2002. The aim of the competition was to build a better understanding between researchers and practitioners of real-life problems by allowing new implementation in the problem introduced. The instances of real-life data represented richer problems and several new requirements and limitations that satisfy the real world implementations in examination timetabling. A description of the problem and the results of the competition for each dataset are available in the competition website at `http://www.cs.qub.ac.uk/itc2007/` (with the exception of the hidden datasets). This dataset is used to investigate the proposed approaches of the present thesis. Further discussion on the survey of the implemented approaches and characteristics of the datasets is presented in Sections 2.3 and 2.5.2.

A recent implementation of a real-world dataset was conducted by Kahar and Kendall (2010) at the Universiti Malaysia Pahang. The problem considered two new real-world constraints, i.e. the distance between examination rooms and separating single examination into several rooms. Since the university is still new and there are few large rooms available, examinations are required to be assigned into multiple rooms. The separation of examinations are evaluated based on the distance between rooms.

2.5 Description of Benchmark Problems

In this thesis, two benchmark datasets - Toronto and ITC2007 - are used to test the proposed approaches. The Toronto dataset is an un-capacitated problem where no room capacity is considered during the timetable construction, while the ITC2007 dataset is a capacitated problem that requires the assignment of examinations to rooms and at the same time satisfies the room capacity restriction. Moreover, the ITC2007 problem is rich with many considerations related to hard and soft constraints. In order to show the generality of the proposed approaches, this thesis explains the implementation relating to both of the benchmark problems. These datasets are described and discussed briefly below.

2.5.1 Toronto

The Toronto instances, also known as 'the Carter benchmark datasets' were introduced by Carter et al. (1996) drawing from various universities around the world and presenting different problem dimensions and characteristics. It consists of thirteen instances which are very well known within the timetabling community. The conflict density of each problem illustrates the difficulty in terms of examinations in conflict. This value is measured by dividing the average number of examinations in conflict with the total number of examinations. However, as stated in Section 2.4, different versions of these datasets circulate within the timetabling community. Qu et al. (2009b) differentiated the various versions of the datasets with new notations. Version I of the introduced datasets is used as the test bed for the proposed approaches. The characteristics of the experimental datasets are summarised in Table 2.3.

Table 2.3: The characteristics of the Toronto benchmark datasets

Problem	Time-slots	Exams	Students	Conflict density
car92	32	543	18 419	0.14
car91	35	682	16 925	0.13
ears83 I	24	190	1 125	0.27
hec92 I	18	81	2 823	0.42
kfu93	20	461	5 349	0.06
lse91	18	381	2 726	0.06
pur93 I	42	2419	30 032	0.03
rye92	23	486	11 483	0.08
sta83 I	13	139	611	0.14
tre92	23	261	4 360	0.18
uta92 I	35	622	21 266	0.13
ute92	10	184	2 750	0.08
yor83 I	21	181	941	0.29

The objective of the Toronto benchmark problem is to create a feasible timetable so that no student is required to sit two examinations at any one time. To achieve a high quality timetable, the soft constraints need to be satisfied as much as possible. Thus, during the timetable construction, it is required that student's examinations are assigned as far apart as possible in order to give a wider student spread in the timetable. The proximity cost function introduced in Carter et al. (1996) in conjunction with the introduced datasets was used in order to measure the quality of the obtained timetable and to describe the average penalty of students distributed in the examination schedule. The formulated cost function is to minimise:

$$\frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} w_{|t_j-t_i|}}{M} \tag{2.1}$$

where N is the number of examinations, c_{ij} is the number of students enrolled to both examinations i and j, t_i is the assigned time-slot for examination i, t_j is the assigned time-slot for examination j, $w_{|t_j-t_i|}$ is the weight whenever students enrolled for two examinations are scheduled $|t_j - t_i|$ apart and M is the total number of students. The penalty weight, $w_{|t_j-t_i|}$ is calculated as $32/2^{|t_j-t_i|}$ where, $|t_j - t_i| \in \{1, 2, 3, 4, 5\}$.

2.5.2 ITC2007

The ITC2007 dataset is used for the evaluation of the approach proposed in this thesis, where the focuses are on the examination timetabling track. It differs from the Toronto datasets in that ITC2007 is a capacitated problem that requires room assignment for each examination. Moreover, time-slot-related constraints and room-related constraints are also considered as the hard constraints to be adhered to. In order to obtain a good quality timetable, several new soft constraints are also taken into account to fulfill the real world requirements; there are seven soft constraints to be satisfied simultaneously with their contribution to the quality of the obtained timetable. The classification of the hard and soft constraints of the examination timetabling track can be viewed in Table 2.4.

The evaluation function of the timetable quality is based on the weighting scheme of several criteria of soft constraint violations derived from the Institutional Model Index. The Institutional Model Index is a weighting system for each soft constraint violation in order to illustrate the quality measure of the timetable that is obtained. For more details on the problem descriptions, the constraints and the mathematical formulations of the examination timetabling track see McCollum et al. (2008) and McCollum et al. (2010). In contrast to other datasets, ITC2007 requires time restrictions in generating the timetable and it is benchmarked using the provided benchmark program as a fair policy in generating solutions. The characteristics of each dataset of the examination timetabling track of ITC2007 is illustrated in Table 2.5.

2.6 Summary

Many ideas have been introduced to the discussion of examination timetabling problems in recent years in order to improve the obtained timetable in terms of solution quality. Within the past two decades, exact methods, approximation algorithms and

Table 2.4: The hard and soft constraints of the examination timetabling track of ITC2007

Constraint type	Descriptions
Hard	- No students should sit two examinations at any one time. - The total number of students in an examination room should not be more than the room capacity and no more than one examinations is allowed to be in the same room. - The examinations are assigned to appropriate time-slot lengths. - Time-slot related constraints (e.g. exam A coincides with exam B) should be satisfied. - Room related constraints (e.g. exam C should be assigned to room 1) should be satisfied.
Soft	- Two examinations in a row: the number of times students sit two adjacent examinations in a day. - Two examinations in a day: the number of times students sitting for examinations in a day. - Period spread: the number of times students sit for examinations within the specified durations. - Mixed durations: the number of times examinations in the same room have a different examination duration. - Larger examinations constraints: the number of examinations with the largest number of enrollment scheduled at the start of the examination session. - Room penalty: the number of times rooms associated with penalty are utilised. - Period penalty: the number of times a time-slot associated with penalty is utilised.

Table 2.5: The characteristic of the ITC2007 benchmark

Problem	Time-slots	Exams	Students	Rooms	Time-slot(HC)	Room (HC)	Conflict density (%)
Exam_1	54	607	7 891	7	12	0	5.05
Exam_2	40	870	12 743	49	12	2	1.17
Exam_3	36	934	16 439	48	170	15	2.62
Exam_4	21	273	5 045	1	40	0	15.0
Exam_5	42	1018	9 253	3	27	0	0.87
Exam_6	16	242	7 909	8	23	0	6.16
Exam_7	80	1096	14 676	15	28	0	1.93
Exam_8	80	598	7 718	8	20	1	4.55
Exam_9	25	169	655	3	10	0	7.84
Exam_10	32	214	1 577	48	58	0	4.97
Exam_11	26	934	16 439	40	170	15	2.62
Exam_12	12	78	1 653	50	9	7	18.45

also constructive heuristic approaches based on graph colouring algorithm have been introduced. Other approaches such as fuzzy-based, decomposition, granular modelling and neural network continue to demonstrate successful implementations regarding the problem. Recently, the most popular of these is the hybrid meta-heuristic which employs search strategy to candidate solution (e.g. simulated annealing, hill climbing, tabu search, great deluge, variable neighbourhood search, large neighbourhood search, developmental approach and harmony search algorithm). Population-based approaches have achieved great success in examination timetabling (e.g. genetic algorithm, memetic algorithm and ant algorithm). Other recent approaches, including case-based reasoning, hyper-heuristic, multi-objective and multi-criteria, also promise to solve examination timetabling problems. Further, the approaches proposed to ITC2007 were presented. The benchmark datasets with various constraints within the examination timetabling problem are also highlighted.

In this thesis, Toronto and the examination timetabling track of ITC2007 benchmark datasets are used as a test bed to determine the superiority of the proposed approaches. These two benchmark datasets can be represented in the form of graph colouring problems in graph theory, thus, making it suitable to construct the initial solutions using the graph colouring heuristics. Furthermore, the constructed solutions can then be improved using approaches, such as meta-heuristics, due to their success in dealing with examination timetabling problems in recent years. Since this problem belongs to the NP-hard problem group (Even et al., 1976), then the exact method is not suitable due to computational expense and impracticality. The overview of approaches described in this chapter provides the ideas for the following chapters on the approaches for solving the examination timetabling problem.

The next chapter discusses the implementation of the proposed approach in constructing a good quality examination timetable.

Chapter 3

Construction of Examination Timetables Based on Adaptive Heuristic Orderings

This chapter presents an initial investigation of the adaptive strategies that order the examinations to be scheduled within a constructive approach. The aim is to construct a good quality timetable based on the ordering and shuffling processes. This study draws on the previous work completed by Burke and Newall (2004) where a heuristic modifier is used to change examination ordering. Examinations are chosen from the ordering based on a shuffling strategy and a stochastic component is incorporated into the process of assigning a chosen examination to a time-slot. Meanwhile, in order to search for a good quality solution, the current best ordering is shuffled so that improved examination ordering could be obtained. Combinations of different graph colouring heuristics during the construction process are also considered by alternating the given heuristics to vary the examination ordering, since different graph colouring heuristics tended to produce different ordering based on the number of unscheduled examinations in the previous timetable construction. The following section presents the algorithm of the adaptive heuristics that order examinations based on priorities inspired by the squeaky wheel optimisation (Joslin and Clements, 1999). Section 3.2 describes the algorithm of un-capacitated and capacitated problems of the implemented datasets. Section 3.3 presents the experiments and discusses the results. Finally, the conclusions are provided in Section 3.4.

3.1 Adaptive Heuristics Ordering the Examinations Based on Priorities

An adaptive approach with a heuristic modifier was applied to the examination timetabling problem by Burke and Newall (2004). Their approach is based on the idea of squeaky wheel optimisation initiated by Joslin and Clements (1999). Squeaky wheel optimisation is a greedy approach and works by iteratively cycling around three procedures: *Constructer*, *Analyzer* and *Prioritizer*. In examination timetabling problems, squeaky wheel optimisation gives priority to a *'difficult'* examination so that it is chosen earlier in the next iteration. In relation to the examination timetabling problem, the procedures are as follows:

- *Constructor*. First, the constructor generates an initial solution for a set of unscheduled examinations based on the initial ordering (which can be generated by a chosen graph colouring heuristic). The unscheduled examinations are individually assigned to the best time-slot i.e. whichever generates the least penalty. During the assignment, there is a possibility that some of the examinations cannot be assigned to a time-slot due to the existence of conflicts with other examinations. In this case, such examinations remain unscheduled.
- *Analyzer*. Once the constructor has completed the assignment, each examination is analysed to check whether there was a problem related with the assignment i.e. whether there is conflict with other examinations during the assignment. A strategy is used to increase the priority of the problematic examination so that it will be given a higher priority in the next iteration. A certain value is added to the difficulty value of the unscheduled examination in order to indicate that this unscheduled examination is more difficult to handle than other examinations. This difficulty value will therefore increase at the end of each iteration if an examination remains unscheduled during the assignment.
- *Prioritizer*. Increasing the difficulty by adding a certain value to a heuristic may change the ordering of examinations. At this stage, the updated difficulty value will be ranked in a decreasing order and the most difficult examination will be chosen to be scheduled first in the next iteration. The process continues until some stopping criterion is met and finally the best solution found is returned.

The proposed approach adapts the examination orderings based on four different graph colouring heuristics. Each examination has a priority determined by the chosen graph coloring heuristic. Such a value can be considered to represent a default difficulty level of scheduling a given examination. If an assignment cannot be found for a certain examination, then it can be considered to be more difficult to schedule than expected. This unscheduled examination is given higher priority in the next iteration. Its difficulty level is modified using a heuristic value added to the value provided by the graph coloring heuristic. The algorithm of the adaptive heuristic ordering approach is explained in Section 3.2.

This approach constructs the solution considering the hard constraint of the tested problem and the quality of the constructed solution is measured based on the soft constraint violation. The type of hard and soft constraints and the evaluation of the quality of the constructed timetable differ across different problem instances. For further detail on the hard and soft constraint evaluation on the tested benchmark datasets see Section 2.5 in Chapter 2.

In order to modify the difficulty value of an examination over time, the idea of a *heuristic modifier* introduced by Burke and Newall (2004) is used. The formula for examination difficulty is presented in equation (3.1). The difficulty of examination i at iteration t is a discrete variable that is an estimation of the difficulty of scheduling the examination after completing the iteration while the heuristic of examination i is a chosen graph colouring heuristic value that estimates the difficulty. $heurmod_i(t)$ for examination i at iteration t is a *heuristic modifier* value. At each iteration, $heurmod_i(t)$ is increased by a *modify* function whenever examination i cannot be scheduled (illustrated in equation (3.1)). This approach can be considered as an online learning algorithm where the feedback from the search process while solving the problem is used to construct the next solution during the iteration.

$$difficulty_i(t) = heuristic_i + heurmod_i(t) \tag{3.1}$$

where,

$$heurmod_i(t{+}1) = \begin{cases} modify(heurmod_i(t)) & \text{, if examination } i \text{ cannot be scheduled} \\ heurmod_i(t) & \text{, otherwise} \end{cases}$$

3.1.1 Graph Colouring Heuristics

Four different graph colouring heuristics were used in this suite of experiments of our study: largest degree (LD), largest enrolment (LE), largest weighted degree (LWD) and saturation degree (SD). The LD, LE and LWD can be categorised as static heuristics because the heuristic value for each examination remains unchanged throughout the iteration. The SD represents a dynamic heuristic due to the dynamic change in each successive examination assignment. One of the objectives of our study is to see the behavior and the performance of these graph colouring heuristics in terms of solution quality and the count of the examinations that violated the hard constraints. In order to compare their contributions to solution quality, a series of experiments has been carried out with different combinations of parameters of these graph colouring heuristics. Below are the details of each graph colouring heuristic:

- *Largest Degree (LD).* The ordering is based on the largest number of conflicting examinations and $heuristic_i(t)$ holds the number of conflicting examinations for examination i. $difficulty_i(t)$ is increased at each of iteration t if the examinations are cannot be scheduled. At this stage, the $heuristic_i(t)$ remains unchanged however, $heurmod_i(t)$ is increased during the iteration based on the modify function. Priority is given to the highest value of difficulty.

- *Largest Enrolment (LE).* The ordering is based on the number of student enrolments for a particular examination where examinations are ordered decreasingly with respect to the heuristic value. $heuristic_i(t)$ holds the number of student enrolments for each examination i at each iteration t.

- *Largest Weighted Degree (LWD).* This heuristic is similar to the largest degree heuristic except that the ordering is based on the number of students in conflict. The $heuristic_i(t)$ holds the number of students in conflict for each examination i at each iteration t. Like other static heuristics, the heuristic value remains unchanged throughout the iteration. Only the difficulty value is increased based on the *heurmod* value.

- *Saturation Degree (SD).* The ordering of examinations is based on the number of remaining time-slots. The examination with the smallest number of available time-slots is scheduled first. The number of remaining time-slots of unscheduled examinations will keep changing as the conflicting examinations are assigned to time-slots. The ordering of unscheduled examinations may change due to the current successive assignment. Since the saturation degree value of an examination decreases from time to time, it requires an adjustment. In this study, the complement of it is used where the saturation degree on an examination is *(max-number-of-time-slot - saturation-degree-of-an-examination).* The saturation degree value, $heuristic_i(t)$ is initialised with 0 and keeps increasing until the maximum number of time-slots is reached if the examination cannot be scheduled during the iteration. The complement of saturation degree is used to increase the difficulty of an examination by adding it to the heuristic modifier. As for the capacitated problem (i.e. the problem relating to room capacity requirement), its saturation degree value also considers the availability of rooms for the remaining time-slot. For example, the number of remaining time-slots of an examination that can be used to schedule is seven. Assuming that three of the remaining time-slots are considered invalid due to the unavailability of rooms. In these circumstances, the number of remaining time-slots of an examination that can be scheduled is reduced to four considering room availability at the same time. Using this heuristic, the priority of choosing an examination is given to the higher value of difficulty. Algorithms 2 and 6 illustrate the process of updating the saturation degree value of the un-capacitated and capacitated problems of the Toronto and ITC2007 benchmark datasets.

3.1.2 Heuristic Modifiers

Different *modify* functions for heuristic modifiers are used in order to express giving priority to the difficult examinations. Equations (3.2), (3.3), (3.4) and (3.5) show the description of each characteristic, where c is a constant and gives different value to the difficulty. The constant value, c is modified to suit the experiments. The modify functions are based on the study by Burke and Newall (2004). For more details, see Section 2.2.3 of the thesis.

- *Custom (C).* This is a conventional strategy of a heuristic. The heuristic modifier has no contribution and the heuristic value solely determines the priority of the examinations to be scheduled. If there are several examinations having the same heuristic value, then a random examination is chosen for scheduling:

$$heurmod_i(t) = heurmod_i(t-1), heurmod_i(0) = 0 \tag{3.2}$$

- *Additive (AD).* The modifier is increased by one at each iteration, if an examination cannot be scheduled. This strategy has a modest effect on the difficulty of a given examination. If the difference between the heuristic value of a given examination i and its predecessor in the priority list is large, then it will take longer in using this approach to reorder the given examination i, emphasizing that this examination i is difficult to schedule:

$$heurmod_i(t) = heurmod_i(t-1) + c, heurmod_i(0) = 0 \tag{3.3}$$

where $c = 1$.

- *Multiplicative (MP).* The modifier value becomes a multiple of a constant where the factor is determined by the current step, c providing a higher priority for the problematic examinations. Reordering for a given examination that is difficult to schedule occurs faster than the AD strategy:

$$heurmod_i(t) = heurmod_i(t-1) + c, heurmod_i(0) = 0 \tag{3.4}$$

where $c = 2$.

- *Exponential (EX).* This modifier will upgrade the priority significantly, if the examination is difficult to schedule:

$$heurmod_i(t) = c \times heurmod_i(t-1), heurmod_i(0) = 1 \tag{3.5}$$

where $c = 2$.

3.1.3 Shuffling the Ordering of Examinations

In order to choose an examination to be scheduled, the examinations are ordered based on the difficulty measure with which graph colouring heuristics are often used. Instead of using the ordering of examinations directly, they can be shuffled within a group of difficult examinations and an unscheduled examination can be chosen based on this shuffling strategy. This strategy uses a *block size* parameter. All ordered examinations are partitioned into blocks of fixed size and are shuffled randomly within each block before an assignment is made. The significance of this strategy is that it gives a different examination ordering by which an examination is chosen randomly from a group that has a close difficulty measure. In these circumstances, the examination to be chosen appears from a certain size of grouped examinations that have been ordered based on graph colouring heuristics.

As an example, Figure 3.1 below illustrates the shuffling strategy within a certain sized block of examinations. Let us say the block size is four. First, all examinations are sorted with respect to their difficulty of scheduling using the chosen graph colouring heuristic and each four consecutive examinations are shuffled within the block randomly. Some of the examinations may remain in the same position because this process is done stochastically. Each examination is scheduled based on this new ordering. It can be noted that this strategy is used only with the static type of graph colouring heuristic. The technique has also been tested with a block size of 0, indicating that the measure(s) used directly determines the difficulty of scheduling an examination for comparison purposes. The experiments are performed using different block sizes in order to observe the effect of this parameter on the performance of the approach. This will be referred to as the *block* approach.

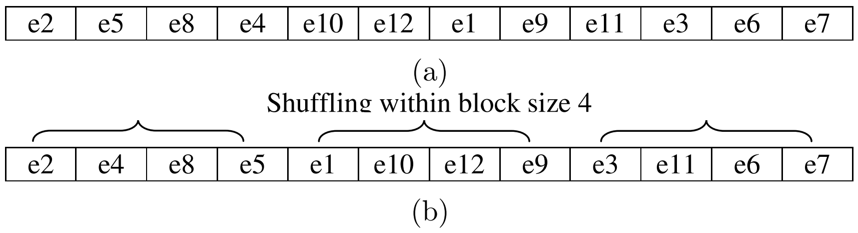

e2 e5 e8 e4 e10 e12 e1 e9 e11 e3 e6 e7
(a)
Shuffling within block size 4
e2 e4 e8 e5 e1 e10 e12 e9 e3 e11 e6 e7
(b)

Figure 3.1: Shuffling strategy within block (a) Ordering of examinations with certain graph colouring heuristic; (b) Ordering after shuffling examinations in the block size 4

For the saturation degree graph colouring heuristic, it is not possible to rearrange the examinations using the block approach due to its dynamic nature. This is because examination ordering keeps changing after each examination assignment, thus making this heuristic unsuitable for block strategy. A previous study (Burke et al., 1998a), suggested that a random examination could be chosen from a fixed number of the most difficult examinations. It can be observed that, this strategy may be used with both static and dynamic heuristics. An examination is chosen randomly from a given number of the most difficult examinations, referred to as *top-window size*. After the selected examination is scheduled, the saturation degree and difficulties are updated accordingly. An example of the shuffling strategy using a saturation degree heuristic is illustrated in Figure 3.2. Let us say the top-window size is four. Once the list of examinations is ordered using the saturation degree heuristic, one examination is chosen randomly from the first four unscheduled examinations. After the chosen examination is assigned to the selected time-slot, the remaining examinations are reordered based on the difficulty of saturation degree values. The next examination to be chosen can be selected among the most difficult examinations within the top-window size four. This strategy will be referred to as *top-window*. The proposed approach is experimented with block and top-window with different sizes in {none, 2, 3, 4, 5, 6, 7, 8, 9} in order to see the difference in shuffling a number of examinations within block or top-window sizes.

3.1.4 Time-slot Choice

Once an examination is chosen, it is assigned to the most appropriate time-slot. The assignment is made to ensure the smallest penalty cost from among all the available time-slot assignments. Previous studies (Burke and Newall, 2004, Burke et al., 2009, Casey and Thompson, 2003) have indicated that the first time-slot that generates the least penalty is chosen for an assignment. Since there is a possibility that some time-slots generate the same least penalty, a random element is incorporated in making this choice, introducing a variation of assignments in the timetable. In such a situation, there is a possibility of an examination being assigned to a different time-slot during another

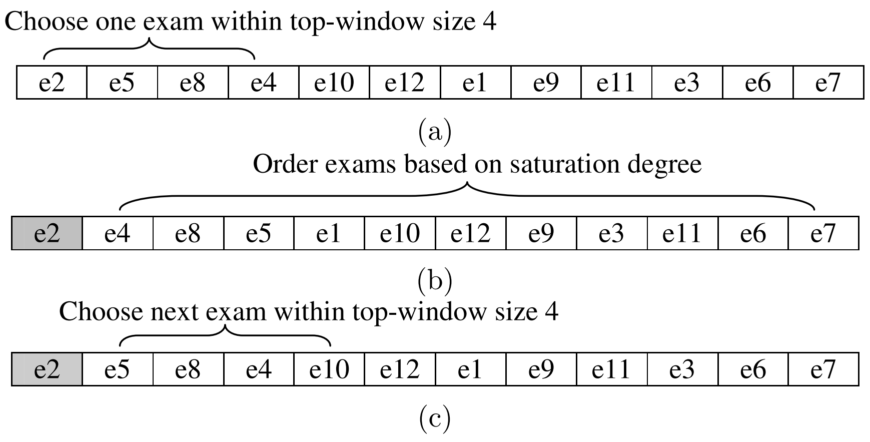

Choose one exam within top-window size 4
e2 e5 e8 e4 e10 e12 e1 e9 e11 e3 e6 e7
(a)
Order exams based on saturation degree
e2 e4 e8 e5 e1 e10 e12 e9 e3 e11 e6 e7
(b)
Choose next exam within top-window size 4
e2 e5 e8 e4 e10 e12 e1 e9 e11 e3 e6 e7
(c)

FIGURE 3.2: Shuffling strategy within top-window size four (a) Ordering of examinations with saturation degree heuristic and choosing one examination from the top-window size four randomly and schedule it; (b) Update the saturation degree of the unscheduled examinations and order them according to difficulty value (no need to update the ordering if using static heuristic); (c) The process continues with choosing the next examination to be scheduled within top-window size four randomly

iteration, even though the order of examinations in the current iteration is the same as in the previous iteration.

3.1.5 Shuffling Best Ordering

The observation on the initial test found that some of the solutions could not be further improved since some of the best solutions were at the early stage of the solution search. It is clear that by allocating more time for the search, there are higher chances of obtaining good results; however, the search needs to be properly established. In these circumstances, whenever there is no improvement to the solution quality for a number of iterations, the solution search is focused on the current best ordering. The examinations are shuffled in the current best ordering using the top-window strategy in order to find a better ordering within the current best solution.

3.1.6 Heuristic Alternation

Different heuristics tend to produce different numbers of violated examinations at each iteration. Observations from the initial tests showed that the saturation degree could produce a smaller number of violated examinations compared with the static graph colouring heuristic due to the dynamic nature of this heuristic. The number of times that the violated examinations occurred in this approach is illustrated by the heuristic modifier, which holds the number of times an examination cannot be scheduled in previous iterations, and the difficulty value is then increased using a modify function.

The higher the value of this heuristic modifier of an examination, the more we have an indication that the examination is more difficult to schedule due to conflict with other examinations in the previous iteration. Since different heuristics give different numbers of violated examinations, it would be advantageous to use different graph colouring heuristics simultaneously during the timetable construction by alternating the heuristics. Different graph colouring heuristics are alternated to order the examinations. Algorithm 7 illustrates the heuristic alternation in constructing the examination timetable.

First, the graph colouring heuristic to be used is chosen randomly. The examination timetable is constructed based on the adaptive heuristic ordering approach and the quality of the obtained timetable is evaluated. Based on the obtained solution quality, the graph colouring heuristic is alternated. In this strategy, when the solution quality is improved, the current graph colouring heuristic is used to construct the examination timetable in the next iteration. Otherwise, if there is no improvement to the solution quality, the current graph colouring heuristic is used for another n trials of timetable construction before proceeding with other graph colouring heuristics. After the n^{th} trial of the current heuristic, if there is still no improvement to the solution quality, then the graph colouring heuristic is alternated with the next graph colouring heuristic that is chosen randomly. The number of trials for this experiment is set as $n = 10$. The aim of the heuristic alternation is to construct the examination timetable differently as the number of violated examinations occurred is varied when incorporating different graph colouring heuristics simultaneously. Using this strategy, the combinations of two, three and all graph colouring heuristics were tested i.e. LD-SD, LD-LE, LD-LWD, SD-LE, SD-LWD, LE-LWD, LD-SD-LE, LD-SD-LWD, LD-LE-LWD, SD-LE-LWD and LD-SD-LE-LWD.

3.2 Algorithm

There are two different benchmark datasets tested using the adaptive heuristic ordering approach, i.e. the un-capacitated problem of the Toronto and the capacitated problem of the ITC2007. The following subsections 3.2.1 and 3.2.2 describe the pseudo-code of the proposed approach for both types of datasets.

3.2.1 Toronto

The pseudo-code of the adaptive heuristic ordering for the un-capacitated problem is illustrated in Algorithm 1. The initial ordering of examinations at the beginning of the timetabling process is set based on the identified graph colouring heuristic. The saturation degree value of each examination i is set to be equal to 0 at the beginning of each iteration if it is chosen as the heuristic of the algorithm. As mentioned in Section 3.1.1, this value will keep increasing up to the maximum number of time-slots and is added with the heuristic modifier indicating the difficulty of certain examinations i. Once the timetabling process begins, the current examination i is checked for the hard constraint violation. If examination i can be scheduled, then it is scheduled with the least penalty time-slot. In the case of more than one same least penalty time-slot available, then the best time-slot is selected randomly from the list. In the case of examination i being violated, then it is left unassigned and at this stage the heuristic modifier of examination i is increased based on the identified type of modify function of the heuristic modifier. For the saturation degree heuristic, the saturation degree value for each examination is updated after each successive examination assignment. After all examinations have been assigned to the time-slots, the solution quality of the constructed timetable is evaluated and the best solution quality is stored.

Algorithm 1 Construction of an examination timetable based on adaptive heuristic orderings for the Toronto benchmark datasets

```
Choose a fixed heuristic and do the initial ordering
for $t = 1$ to number of iterations do
  if Saturation_degree then
    Set the saturation degree for each examination as 0
  end if
  for $i = 1$ to number of examinations do
    Apply shuffling strategy (block or top-window)
    if $i$ can be scheduled then
      Schedule $i$ in the time-slot with the least penalty. In the case of the availability
      of multiple time-slots with the same penalty, choose one randomly
    else
      Increase and modify heuristic modifier of $i$
    end if
    if Saturation_degree then
      Update the Saturation_degree (Algorithm 2)
    end if
  end for
  Evaluate solution, store if it is the best found so far
end for
```

Algorithm 2 shows how the saturation degree for each examination is changed during the timetable construction of the un-capacitated problem. The saturation degree for each unscheduled examination is updated considering the current assignment by checking their conflict. If the examination to be scheduled, s conflicts with the current scheduled examination i and the current time-slot has not yet been assigned to any conflicting examination, then the saturation degree of the unscheduled examinations s is increased. Algorithms 1 and 2 are applied to the Toronto benchmark datasets while Algorithm 7 is employed for the heuristic alternation strategy. This has also been explained in Section 3.1.6.

Algorithm 2 Saturation degree for the un-capacitated problem

```
for each examination s that is not scheduled yet do
  if $s$ is in conflict with $i$ then
    Increase saturation degree value until number of time-slots
  end if
end for
```

3.2.2 ITC2007

The ITC2007 benchmark datasets represent heavily constrained problems and require different treatment in order to satisfy the feasibility of each problem. Since these datasets are different from the Toronto in terms of the room constraints and the existence of a number of other hard and soft constraints, it is necessary to code them differently. Nevertheless, the general framework of the adaptive heuristic ordering approach remains the same. The pseudo-code of the adaptive heuristic ordering of the ITC2007 benchmark datasets is illustrated in Algorithms 3, 4, 5 and 6. Algorithm 3 illustrates how the approach works for the ITC2007 benchmark datasets which consider the time-slots and room requirements simultaneously. During the time-slots search of examination i, two types of time-slot lists are considered. The first is the list of time-slots with the same least penalty, named as '(All_best_slot)' list where the penalty is calculated based on the time-slot utilisation. The second is the list of time-slots that are not the least penalty but not violated by examination i, named as '(All_feasible_slot)' list. The feasible list is used whenever the least penalty time-slot fails to create a feasible assignment when considering room availability or due to the violation to other pairs of examinations related with hard constraint requirement. Scheduling of examinations of the hard constraint requirement is essential in order to achieve feasibility for the obtained timetable. For further discussion on the hard and soft constraints of the ITC2007 benchmark datasets see Section 2.4. In general, the partitioning into two sets is also applied to the room search, i.e. a list of same least penalty rooms, '(All_best_room)' and feasible rooms, '(All_feasible_room)'.

In these benchmark datasets, the heuristic modifier of examination i has the chance to be increased twice during the iteration since there is a possibility that examination i cannot be scheduled to a time-slot and a room. If examination i can be assigned to a certain time-slot but no room is available, despite a number of assignment trials, then the heuristic modifier of examination i is increased once during the iteration. During the timetabling process, if examination i is violated due to room assignment, then the reassignment process is started (as illustrated in Algorithm 5). Examination i is unassigned from the current best time-slot and one new best time-slot j within the (All_best_slot) list will be assigned to examination i and the search for a new best room will continue. The search for the best room for examination i continues with the new best time-slot j from the (All_best_slot) list, or the (All_feasible_slot) is used if no feasibility is achieved from the (All_best_slot). If there is still no room available for examination i, then the heuristic modifier for examination i is increased.

During the timetable construction, examination i is verified whether this examination is one of the examinations from the hard constraint list. The examinations in the hard constraint list are required to be satisfied and they are related with the time-slot and room related hard constraints. For more details on the constraints of the ITC2007 see Section 2.4. If so, then this examination has to be assigned to the appropriate time-slot and room simultaneously with the related examination(s). Algorithm 4 shows the assignment process of the examinations related to hard constraints. Once examination i is identified as one of the examinations from the hard constraint list, the assignment of examination i to a time-slot and room is verified to see whether it suits the hard constraint requirement that relates to its pair i.e. examination h. At the same time, the best time-slot and room for examination h are identified and they are assigned to examination h if it fulfills the requirement. If searches for a non-violated time-slot and a room for examination h are unsuccessful, then a reassignment process is started. This is may be because the time-slot or room assignment of examination i has prompted examination h to create a hard constraint violation. The time-slot and room search of examination h may be unsuccessful due to the assignment of examination i. For example, supposed examination i should be assigned to the same time-slot with examination h, or if examination i should be concurrent with examination h; however, the assignment cannot be made because examination h is violated the best time-slot of examination i and cannot be assigned to this best time-slot. In this case, the best time-slot of examination i should be changed to another time-slot in order to ensure that examination h could be assigned to the same time-slot as examination i. If all the time-slots of examination i in the (All_best_slot) list violated examination h, then these are kept tabu so that these time-slots are not chosen again in the next assignment trial. Next, the new best time-slot of examination i is obtained by choosing one time-slot randomly from the (All_feasible_slot) list. After the reassignment process has been repeated for a number of times and the examination h still cannot be assigned to any time-slot or room, then this examination h is kept unscheduled.

With this approach, the saturation degree of the capacitated problem is treated differently. Since the room requirement should be satisfied, the saturation degree value of

Algorithm 3 Construction of an examination timetable based on adaptive heuristic orderings for the ITC2007 benchmark datasets

```
Choose a fixed heuristic and do the initial ordering
for $t = 1$ to number of iterations do
  if Saturation_degree then
    Set the saturation degree for each examination as 0
  end if
  for $i = 1$ to number of examinations do
    Apply shuffling strategy (block or top window)
    //Find time-slot for examination $i$
    for $j = 1$ to number of time-slots do
      Find (All_best_slot) and (All_feasible_slot)
    end for
    if $i$ can be scheduled to any time-slot then
      Choose one best_slot $j$ randomly from (All_best_slot)
      Assign $i$ to best_slot $j$
    else
      Increase and modify heuristic modifier of $i$
    end if
    //Find rooms related to best_slot $j$
    for $k = 1$ to number of rooms do
      Find (All_best_room) and (All_feasible_room)
    end for
    if $i$ can be scheduled to room then
      Choose best_room randomly from (All_best_room)
      Assign $i$ to best_room
    else
      if more time-slot from the (All_best_slot) is available then
        Reassignment Process (Algorithm 5)
      else
        Choose new time-slot $j$ from the (All_feasible_slot)
        Reassignment Process (Algorithm 5)
      end if
    else
      if No room for (All_best_slot) and (All_feasible_slot) then
        Increase and modify heuristic modifier of $i$
      end if
    end if
    if $i$ is examination in the hard constraint list then
      Do the assignment of examination(s) related to $i$ (Algorithm 4)
    end if
    if Saturation_degree then
      Update the Saturation_degree of $i$ (Algorithm 6)
      if $i$ is examination in the hard constraint list then
        Update saturation degree of related examinations (Algorithm 6)
        Update difficulty of related examinations in the hard constraint list
      end if
    end if
  end for
  Evaluate solution, store if it is the best found so far
end for
```

Algorithm 4 Examination assignment related to hard constraints requirements

```
for $h = 1$ to number of examinations hard constraint do
  if $i$ is related to $h$ then
    Find time-slot and room of $h$ and assign them
    while $h$ is violated do
      Reassignment process of hard constraint violation:
      Unassigned and change time-slot of $i$ from (All_best_slot)
      while time-slot of (All_best_slot) of $i$ violated to $h$ do
        Choose other time-slot randomly from (All_best_slot)
        if all time-slot of (All_best_slot) has been checked and violated to $h$ then
          tabu (All_best_slot)
          while time-slot of (All_feasible_slot) of $i$ violated to $h$ do
            Choose other time-slot from (All_feasible_slot)
            if all time-slot of (All_feasible_slot) has been checked and violated to $h$
            then
              Stop the search
              Increase and modify heuristic modifier of $h$
            end if
          end while
        end if
      end while
    end while
  end if
end for
```

Algorithm 5 Reassignment process

```
Unassign $i$ and choose a new time-slot from the list
while no feasible room for the chosen time-slot do
  Choose other time-slot from the list
  if all time-slots in the list have been checked and no feasible room available then
    Stop the search and tabu the list
  end if
end while
```

each unscheduled examination i is also updated according to room requirement. In Algorithm 6, once the saturation degree of each of the unscheduled examinations i have been updated, each non-violated time-slot is checked for room availability. Room availability consists of rooms that fulfill the capacity requirement, i.e. the number of students for a certain examination is less or equal to the room capacity. If the non-violated time-slot has no room to be assigned to it, then the saturation degree of examination i is updated accordingly. On the other hand, the largest degree heuristic is treated the same as in the un-capacitated problem where it does not consider the room capacity requirement during each examination assignment. Algorithm 7 illustrates the heuristic alternation of the adaptive heuristic ordering and has been explained in Section 3.1.6.

Algorithm 6 Saturation degree for the capacitated problem

```
for each examination $s$ that is not scheduled yet do
  if $s$ is in conflict with $i$ then
    Increase saturation degree value of $s$ until number of time-slots
    for each feasible time-slot $fs$ of examination $s$ do
      if $fs$ has no feasible room then
        Increase saturation degree value of $s$ until number of time-slots
      end if
    end for
  end if
end for
```

Algorithm 7 Alternating the heuristics during the search

```
$G = \{SD, LD, LE, LWD\}$
Set number of trial, $n$
Choose heuristics randomly from $G$
for $t = 1$ to number of iterations do
  if Saturation_degree then
    Set the saturation degree for each examination as 0
  end if
  Timetable construction based on adaptive heuristic ordering (Algorithm 1 for Toronto and Algorithm 3 for ITC2007 benchmark datasets).
  Evaluate solution, store if it is the best found so far
  if Solution quality is improved then
    Continue with the current heuristic
  else
    Count number of trial
    if number of trial is more than $n$ then
      Randomly alternate to other heuristic in $G$
    end if
  end if
end for
```

3.3 Experimental Results

The experiments were conducted on a PC with Intel Harpertown 3.0 GHz. processor and 16 Gb memory. All runs were repeated fifty times for the Toronto and ITC2007 benchmark datasets. The solution was generated for each combination of graph coloring heuristic, heuristic modifier, shuffling strategy and the relevant parameter due to the stochastic nature of the proposed approaches. Each run of the Toronto benchmark dataset was terminated whenever the maximum number of iterations was reached. Two different values, $\{2000, 4000\}$ were used for the maximum number of iterations during the experiments for the Toronto benchmark datasets while the ITC2007 benchmark datasets followed the running time requirement as stated in the competition rules. The details of the ITC2007 can be found at `http://www.cs.qub.ac.uk/itc2007/`. Sections 3.3.1 and

3.3.2 discuss the experimental results of the proposed approaches of the Toronto and ITC2007 benchmark datasets, respectively. The best result for each problem instance is highlighted in bold font. Section 3.3.3 discusses the analysis of the approaches. For the rest of the discussion, the proposed approach is represented as adaptive heuristic ordering (AHO).

3.3.1 Toronto

Table 3.1 illustrates the experimental results for the four graph colouring heuristics using different combinations of algorithmic choices, respectively using the basic AHO approach. The basic AHO approach consists of the implementation of different combinations of heuristic modifier and block or top-window strategy but with no shuffling best ordering. The table reports the best penalty values obtained out of fifty runs for the four graph colouring heuristics for each combination and each problem instance. A saturation degree-based approach provides the best performance compared with other graph colouring-based approaches in most of the problem instances, except for lse91 and pur93 I. The best solution quality for lse91 is obtained when using the largest enrolment heuristic, while for pur93 I, it is achieved by using the largest degree heuristic. Considering the maximum number of iterations, the saturation degree with 4000 iterations performs better than 2000 iterations by producing eight best results out of thirteen instances.

As demonstrated in Table 3.1, the best results for the saturation degree heuristic are mostly obtained by using the exponential modifier (six best results), showing that upgrading the modifier with large values could yield a better ordering of examinations. This approach is followed by the custom modifier approach, which does not make use of a heuristic modifier, achieving four best results. The difference in the custom approach from previous implementations is that the AHO approach utilised the idea of assigning a random time-slot in case of equal quality possibilities for a given unscheduled examination, and the top-window strategy is also incorporated. The multiplicative and additive modifiers obtain two and one best results respectively, for the saturation degree heuristic.

In considering the largest degree graph colouring heuristic, Table 3.1 shows that the exponential heuristic modifier is the best choice for changing the order of examinations based on its difficulty. The exponential heuristic modifier provides ten best results out of thirteen instances, followed by the multiplicative heuristic modifier with three best results. The custom and additive heuristic modifiers do not deliver a good performance since they made small changes in updating the difficulty value and take a greater amount

TABLE 3.1: Comparison of different heuristics with different combinations of algorithmic choices of the basic AHO for Toronto benchmark datasets (LD = largest degree, LE, largest enrolment, LWD = largest weighted degree, SD = saturation degree)

	Combination of algorithmic choices {number of iterations, modifier type, block/top-window size}			
Problem	LD	LE	LWD	SD
car91	5.36 {4000, EX, 9}	5.51 {4000, AD, 5}	5.66 {2000, AD, 8}	**5.08** {4000, EX, 5}
car92	4.56 {4000, EX, 3}	4.55 {2000, EX, none}	4.80 {4000, MP, 3}	**4.38** {2000, EX, none}
ears83 I	40.00 {4000, MP, 3}	39.83 {2000, AD, 5}	40.48 {4000, MP, 9}	**38.44** {4000, MP, 2}
hec92 I	11.84 {2000, MP, 6}	11.93 {4000, MP, 5}	11.55 {2000, MP, 5}	**11.61** {2000, C, 5}
kfu93	15.54 {4000, EX, none}	14.86 {4000, C, 2}	15.65 {2000, AD, 9}	**14.67** {4000, EX, 2}
lse91	11.78 {4000, EX, 3}	**11.67** {2000, AD, 6}	12.15 {4000, MP, 2}	11.69 {2000, MP, 6}
pur93 I	**5.91** {4000, MP, 9}	6.20 {2000, AD, 2}	6.49 {4000, AD, 5}	5.93 {2000, EX, 8}
rye92	9.69 {4000, EX, 4}	9.94 {4000, EX, 6}	10.09 {4000, MP, 5}	**9.49** {4000, AD, 5}
sta83 I	157.85 {4000, EX, 9}	158.19 {4000, AD, 7}	157.82 {4000, AD, 4}	**157.72** {4000, C, none}
tre92	8.88 {4000, EX, 2	9.08 {4000, EX, 3}	9.12 {4000, MP, 4}	**8.78** {4000, C, 9}
uta92 I	3.66 {4000, EX, 2}	3.70 {4000, AD, 3}	3.75 {4000, EX, 3}	**3.55** {4000, EX, 3}
ute92	26.82 {4000, EX, 7}	27.37 {4000, AD, 7}	27.36 {4000, MP, 6}	**26.63** {2000, EX, 7}
yor83 I	41.59 {4000, EX, 6}	41.99 {4000, MP, 3}	43.75 {2000, AD, 7}	**40.45** {4000, C, 5}

of time to demonstrate significant changes to the examinations ordering. On the other hand, the largest enrolment heuristic works effectively with the additive heuristic modifier, which provides seven out of thirteen problem instances, followed by the exponential heuristic modifier providing best results for three instances, and the multiplicative and custom modifiers providing best results for two and one problem instances respectively. Meanwhile, the largest weighted degree graph colouring heuristic performs well with the multiplicative heuristic modifier, on seven of the instances, followed by the additive heuristic modifier with five instances and the exponential heuristic modifier with one instance. It can be concluded that, in most cases, the approach works effectively by obtaining more best results when more searching time is given, i.e. with 4000 as the maximum number of iterations as experimented in this study.

It can also be concluded that the block or top-window size choice affects the performance on the shuffling strategy especially with the smaller block/top-window size. However, as can be seen in Table 3.1, increasing the block or top-window to a larger size does not seem to improve the performance significantly. This might be because the arrangement of examinations in bigger chunks reduces the effectiveness of the approach by increasing the chance of shifting towards a more random ordering of examinations. As another approach using a graph colouring heuristic, different parametric choices using a number of block and top-window sizes respectively are considered to execute. Since the parametric choice in both cases is a constant factor (2 to 9), it does not affect the overall running time. The discussion on the statistical test on the size of block and top-window for the Toronto and ITC2007 benchmark datasets is discussed in Section 3.3.3.

The solution quality of the implementation of basic AHO and AHO with shuffling best ordering (as described in Section 3.1.5) for four different graph colouring heuristics is illustrated in Table 3.2. As can be seen, most of the time, the largest degree heuristic of AHO with shuffling best ordering can improve the solution quality compared with the largest degree heuristic of the basic AHO, where ten out of thirteen instances obtain better results. The saturation degree heuristic of the AHO with shuffling best ordering obtains four better results, while the largest enrolment heuristic and the largest weighted degree heuristic achieve six and five better results respectively, compared with the basic AHO. Nevertheless, for five problems, (i.e. hec92 I, lse91, pur93 I, sta83 I and ute92) the AHO with the shuffling strategy is overall better than the basic AHO. However, although the incorporation of shuffling best ordering strategy with the basic AHO can improve the solution quality, the results in Table 3.2 show that the improvement rate is not very significant however the running time is about the same for both approaches.

The heuristic alternation strategy of AHO, illustrated in Table 3.3, also produces comparable results. Heuristic alternation is better than the basic AHO and the AHO with the

Table 3.2: Comparison of basic AHO and AHO with shuffling best ordering for four different graph colouring heuristics for the Toronto benchmark datasets (LD = largest degree, SD = saturation degree, LE = largest enrolment, LWD = largest weighted degree, t(s) = running time in seconds)

	Basic AHO				AHO - shuffling best ordering			
Problem	LD	LE	LWD	SD	LD	LE	LWD	SD
car91	5.36	5.51	5.66	**5.08**	5.33	5.51	5.72	5.17
t(s)	35	92	37	145	77	77	36	162
car92	4.56	4.55	4.80	**4.34**	4.57	4.61	4.80	4.45
t(s)	22	28	48	164	48	27	49	183
ears83 I	40.0	39.83	40.48	**38.44**	39.01	40.15	40.61	39.35
t(s)	4	5	9	10	5	6	10	12
hec92 I	11.78	11.93	11.55	11.61	11.41	11.87	11.37	**11.13**
t(s)	1	2	1	3	1	2	2	2
kfu93	15.54	14.86	15.65	**14.67**	14.92	15.43	15.32	15.0
t(s)	6	12	6	105	13	12	12	140
lse91	11.78	11.67	12.15	11.69	**11.56**	12.0	12.25	11.79
t(s)	4	5	10	44	9	11	4	66
pur93 I	5.91	6.20	6.49	5.93	**5.90**	6.18	6.50	5.91
t(s)	146	229	300	1803	148	317	210	2741
rye92	9.69	9.94	10.09	**9.49**	9.78	9.83	10.05	9.67
t(s)	10	23	20	74	23	20	20	66
sta83 I	157.85	158.19	157.82	157.72	**157.34**	157.76	158.09	157.58
t(s)	1	3	2	9	2	2	1	11
tre92	8.88	9.08	9.12	**8.78**	8.94	8.83	9.20	8.84
t(s)	5	11	12	17	11	13	11	20
uta92 I	3.66	3.70	3.75	**3.55**	3.61	3.72	3.77	3.56
t(s)	28	64	75	134	60	32	60	222
ute92	26.82	27.37	27.36	26.63	26.55	27.02	27.35	**26.40**
t(s)	1	2	2	10	3	2	2	7
yor83 I	41.59	41.99	43.76	**40.45**	40.76	43.19	43.23	40.48
t(s)	3	7	5	17	3	10	8	23

shuffling best ordering approach. However, this does not apply to the saturation degree heuristic because in most cases the saturation degree heuristic of the basic AHO approach performs better than the heuristic alternation strategy. As the table demonstrates, in most problems the combination of the largest degree heuristic or the saturation degree heuristic with the largest enrolment heuristic has the potential of obtaining a good solution quality. The results of combining two heuristics show that each of the combination of LD-LE and SD-LE obtain four best results out of thirteen instances. The combination of three heuristics, LD-SD-LE, LD-LE-LWD and SD-LE-LWD, obtained four, three and five best results respectively. This indicates that the combination of graph colouring heuritics during timetable construction is very useful in helping to order the examinations based on their difficulties while at the same time producing good solution quality

Table 3.3: Comparison of different combinations of graph colouring heuristics in heuristic alternation strategy of AHO for the Toronto benchmark datasets (The bold entries indicate the best results for given heuristic combination type, while those in bold and italic indicate the best results found for the given problem) (LD = largest degree, SD = saturation degree, LE = largest enrolment, LWD = largest weighted degree)

Problem	Heuristic combinations										
	Two heuristics						Three heuristics				All heuristics
	LD-SD	LD-LE	LD-LWD	SD-LE	SD-LWD	LE-LWD	LD-SD-LE	LD-SD-LWD	LD-LE-LWD	SD-LE-LWD	LD-SD-LE-LWD
car91	5.16	5.21	5.3	5.19	**5.15**	5.36	5.18	***5.11***	5.29	5.13	5.18
t(s)	213	75	85	299	96	52	83	108	84	79	75
car92	4.49	4.44	4.59	4.44	4.51	**4.42**	4.41	4.5	4.47	***4.39***	4.43
t(s)	24	58	54	69	67	28	58	65	25	180	93
ear83 I	39.07	38.99	38.94	39.01	***38.28***	39.63	**38.98**	39.85	39.55	39.31	39.18
t(s)	9	9	5	8	10	13	15	8	14	8	6
hec92 I	11.43	11.57	11.32	***11.23***	11.32	11.49	11.52	11.56	11.53	**11.39**	11.45
t(s)	3	1	3	2	3	2	2	3	2	3	3
kfu93	14.96	14.81	15.06	14.63	***14.42***	14.9	14.97	14.9	**14.82**	14.83	14.9
t(s)	37	6	14	83	77	14	54	78	13	62	55
lse91	**11.62**	**11.62**	11.7	11.66	11.79	11.72	***11.43***	11.92	**11.64**	11.65	11.66
t(s)	39	4	9	48	40	11	17	35	9	14	15
pur93 I	5.86	***5.74***	5.93	5.85	5.9	5.85	5.82	5.94	**5.79**	5.87	5.87
t(s)	1508	346	191	1905	1543	394	2910	1173	162	1158	297
rye93	**9.56**	9.6	9.58	9.59	9.63	9.65	**9.54**	9.57	9.57	9.57	***9.37***
t(s)	70	22	21	80	37	21	93	57	10	60	49
sta83 I	157.55	**157.53**	157.84	157.62	157.55	157.59	**157.68**	157.8	***157.45***	157.7	157.69
t(s)	6	1	2	4	8	3	2	4	3	4	5
tre92	**8.83**	8.96	8.92	***8.73***	8.86	8.88	8.9	8.9	8.95	**8.74**	8.91
t(s)	31	6	11	12	27	14	26	28	11	11	10
uta92 I	3.59	**3.53**	3.66	***3.52***	3.64	3.54	**3.55**	3.6	3.58	3.56	3.57
t(s)	84	60	30	76	102	43	195	189	61	146	129
ute92	26.43	**26.3**	26.73	26.78	26.76	26.67	**26.27**	26.64	26.65	26.55	***26.24***
t(s)	5	2	1	7	12	1	7	6	2	8	3
yor83 I	40.63	40.82	41.92	***40.38***	40.87	41.6	40.66	40.62	41.54	**40.49**	41.15
t(s)	14	4	7	8	9	10	17	6	9	15	11

compared with using only a single graph colouring heuristic.

Tables 3.4, 3.5 and 3.6 illustrate the best results obtained for the Toronto benchmark datasets as reported in the literature, using constructive, hyper-heuristics and improvement approaches respectively, compared with the best results of AHO approaches. A comparison with previously proposed constructive approaches in Table 3.4 reveals that the AHO approach provides no best result. Nevertheless, the AHO approach is competitive with other previous constructive approaches in the literature. Overall, the results obtained are comparable, and are very close, to the best known among constructive approaches - for example, car91, car92, sta83 I, uta92 I and yor83 I. The AHO approach generates a better performance in five out of thirteen instances, i.e. car91, car92, sta83 I, tre92 and yor83 I, compared with the previous approaches described by Carter and Laporte (1996). A comparison with the constructive approach proposed by Burke and Newall (2004) shows that the AHO approach generates better results for five instances, namely hec92 I, kfu93, sta83 I, ute92 and yor83 I. Additionally, the approach by Burke and Newall (2004) does not provide results for rye92 and pur93 I. However, in terms of the running time, the approach by Burke and Newall (2004) is faster than our approach. This is due to the process of finding the least penalty time-slot that considered all time-slots to be evaluated and also the incorporation of the shuffling strategy.

The comparison with previous hyper-heuristic approaches depicted in Table 3.5 shows that the AHO approach generates best results for four instances (hec92 I, kfu93, rye92 and ute92) out of thirteen of the Toronto benchmark datasets. The AHO approach obtains good solution quality even though no further improvement is required, while most of the hyper-heuristic approaches incorporated improvement strategy. As illustrated in Table 3.6, the best results obtained using improvement approaches are stronger, but the results of the AHO approach is nevertheless comparable with the previous improvement strategies.

3.3.2 ITC2007

Table 3.7 shows the results of basic AHO implemented in the ITC2007 benchmark datasets with different combinations of algorithmic choices for largest degree, largest enrolment, largest weighted degree and saturation degree heuristics. As can be seen, the largest degree heuristic produces a better performance in terms of the solution quality compared with other heuristics, producing eight better results out of the twelve instances. This differs from the Toronto benchmark datasets where the saturation degree approach is better than the other heuristics. The saturation degree heuristic obtains two best results while the largest enrolment and the largest weighted degree heuristic each

Table 3.4: Comparison of AHO with different constructive approaches of the Toronto benchmark datasets

Problem	[1]	[2]	[3]	[4]	[5]	[6]	AHO
car91	7.10	**4.97**	5.45	5.29	5.03	5.18	5.08
car92	6.20	4.32	4.50	4.54	**4.22**	4.44	4.34
ears83 I	36.40	36.16	36.15	37.02	**36.06**	39.55	38.28
hec92 I	**10.80**	11.61	11.38	11.78	11.71	12.20	11.13
kfu93	**14.00**	15.02	14.74	15.80	16.02	15.46	14.42
lse91	**10.50**	10.96	10.85	12.09	11.15	11.83	11.43
pur93 I	**3.90**	-	-	-	-	4.93	5.74
rye92	**7.30**	-	-	10.38	9.42	10.04	9.37
sta83 I	161.50	161.9	**157.21**	160.4	158.86	160.50	157.34
tre92	9.60	8.38	8.79	8.67	**8.37**	8.71	8.73
uta92 I	3.50	**3.36**	3.55	3.57	3.37	3.49	3.52
ute92	**25.80**	27.41	26.68	28.07	27.99	29.44	26.24
yor83 I	41.70	40.77	42.20	39.8	**39.53**	42.19	40.38

[1]-Carter and Laporte (1996); [2]-Burke and Newall (2004); [3]-Qu and Burke (2007); [4]-Asmuni et al. (2009); [5]-Burke et al. (2010e); [6]-Pais and Burke (2010)

Table 3.5: Comparison of AHO with different hyper-heuristics approaches of the Toronto benchmark datasets

Problem	[7]	[8]	[9]	[10]	[11]	[12]	AHO
car91	5.37	5.36	**4.97**	5.16	5.17	5.19	5.08
car92	4.67	4.53	4.28	**4.16**	4.32	4.31	4.34
ears83 I	40.18	37.92	36.86	35.86	**35.70**	35.79	38.28
hec92 I	11.86	12.25	11.85	11.94	11.93	11.19	**11.13**
kfu93	15.84	15.20	14.62	14.79	15.30	14.51	**14.42**
lse91	-	11.33	11.14	11.15	11.45	**10.92**	11.43
pur93 I	-	-	**4.73**	-	-	-	5.74
rye92	-	-	9.65	-	-	-	**9.37**
sta83 I	157.38	158.19	158.33	159.00	159.05	**157.18**	157.34
tre92	**8.39**	8.92	8.48	8.60	8.68	8.49	8.73
uta92 I	-	3.88	3.40	3.42	**3.30**	3.44	3.52
ute92	27.60	28.01	28.88	28.30	28.00	26.70	**26.24**
yor83 I	-	41.37	40.74	40.24	40.79	**39.47**	40.38

[7]-Kendall and Mohd Hussin (2005a); [8]-Burke et al. (2007); [9]-Pillay and Banzhaf (2009); [10]-Qu and Burke (2009); [11]-Qu et al. (2009a); [12]-Burke et al. (2010f)

Table 3.6: Comparison of AHO with different improvement approaches of the Toronto benchmark datasets

Problem	[13]	[14]	[15]	[16]	[17]	[18]	[19]	AHO
car91	5.1	**4.5**	5.4	5.2	6.6	4.6	4.8	5.08
car92	4.3	3.93	4.2	4.4	6.0	**3.90**	4.1	4.34
ears83 I	35.1	33.71	34.2	34.9	**29.30**	32.8	34.92	38.28
hec92 I	10.6	10.83	10.4	10.3	**9.2**	10.0	10.73	11.13
kfu93	13.5	13.82	14.3	13.5	13.8	**13.0**	**13.0**	14.42
lse91	10.5	10.35	11.3	10.2	**9.6**	10.0	10.01	11.43
pur93 I	-	-	-	-	**3.7**	-	4.73	5.74
rye92	8.4	8.53	8.8	8.7	**6.8**	-	9.65	9.37
sta83 I	157.3	158.35	157.0	159.2	158.2	**156.9**	158.26	157.34
tre92	8.4	7.92	8.6	8.4	9.4	7.9	**7.88**	8.73
uta92 I	3.5	**3.14**	3.2	3.6	3.5	3.2	3.2	3.52
ute92	25.1	25.39	25.3	26.0	**24.4**	24.8	26.11	26.24
yor83 I	37.4	36.53	36.4	36.2	36.2	**34.9**	36.22	40.38

[13]-Merlot et al. (2003); [14]-Yang and Petrovic (2004); [15]-Côté et al. (2005); [16]-Abdullah et al. (2007); [17]-Caramia et al. (2008); [18]-Burke et al. (2010a); [19]-Turabieh and Abdullah (2011).

achieves one best result for the basic AHO. There is also an overall variation in the type of heuristic modifier. The custom approach with no modifier does not perform well for these datasets, while the largest enrolment and the largest weighted degree fail to generate any feasible solution for Exam_4.

Table 3.7 also illustrates that the largest degree heuristic performs well with the exponential heuristic modifier providing best results for six instances, the multiplicative and additive heuristic modifiers each for three instances and no best result for the custom heuristic modifier. The saturation degree heuristic also delivers variation in the performance of heuristic modifiers. The results show that each of the exponential, multiplicative and additive heuristic modifiers obtain four best results while there are none for the custom heuristic modifier.

The exponential heuristic modifier shows great success when implemented in the largest weighted degree heuristic, with eight of the tested instances performing best with the exponential heuristic modifier, the multiplicative and additive heuristic modifiers produced two best results respectively, while none for the custom heuristic modifier. Meanwhile, the largest enrolment heuristic has variation in performance within the type of heuristic modifier. The largest enrolment heuristic performs best for five problems with the multiplicative heuristic modifier, four problems with the additive heuristic modifier, three problems with the exponential heuristic modifier and none for the custom heuristic modifier.

Table 3.7: Comparison of different heuristics with different combination of algorithmic choices of the ITC2007 benchmark datasets for basic AHO (LD = largest degree, LE = largest enrolment, LWD = largest weighted degree, SD = saturation degree, *inf.* = infeasible).

	Combination of algorithmic choices {modifier type,block/top-window size}			
Problem	LD	LE	LWD	SD
Exam_1	**11155** {AD, 9}	12261 {AD, 7}	11897 {EX, 8}	11228 {AD, 5}
Exam_2	3272 {EX, 4}	3175 {EX, 6}	**3128** {EX, 6}	3159 {EX, 0}
Exam_3	**19661** {EX, 9}	20957 {AD, 7}	21204 {EX, 9}	19705 {MP, 8}
Exam_4	**23978** {MP, 3}	*inf.*	*inf.*	24195 {AD, 2}
Exam_5	8033 {EX, 6}	8768 {MP, 0}	8678 {EX, 4}	**8032** {AD, 7}
Exam_6	**28295** {EX, 2}	58445 {MP, 5}	28425 {EX, 4}	28580 {MP, 5}
Exam_7	16002 {EX, 2}	**15573** {MP, 9}	16115 {MP, 9}	16163 {AD, 6}
Exam_8	**19802** {MP, 7}	20322 {AD, 6}	20124 {EX, 3}	20146 {EX, 7}
Exam_9	**2192** {AD, 9}	2207 {AD, 2}	2248 {EX, 9}	2271 {EX, 0}
Exam_10	**16828** {EX, 4}	17429 {MP, 8}	17301 {AD, 8}	17336 {MP, 6}
Exam_11	**44921** {MP, 3}	48807 {EX, 2}	49482 {EX, 4}	46583 {EX, 0}
Exam_12	6502 {AD, 6}	6654 {MP, 4}	6755 {AD, 2}	**5858** {MP, 5}

A comparison of basic AHO and AHO with shuffling best ordering of four different graph colouring heuristics is presented in Table 3.8. It is interesting to see that some of the heuristics fail to improve the solution quality even though the shuffling strategy is incorporated. Some of the solutions do not improve when compared with the basic AHO. Further, the improvements to some of the problem instances are not so significant because only small increments occurred to the solution quality. This may be due to the nature of the problem to be solved since it is a complex problem and involves many constraints.

Table 3.9 illustrates the AHO with different combinations of graph colouring heuristics within the heuristic alternation strategy. Considering various heuristics during the timetable construction appear to assist better ordering, especially in improving the solution quality of the single heuristics of largest enrolment and largest weighted degree. Nevertheless, this is not the case for the largest degree and saturation degree heuristics, where the improvements to their solution quality are not very significant. A consideration of the combination of largest degree and largest enrolment heuristics shows that it is a success where it obtains five best results compared with other combinations of two heuristics. It is also demonstrated in the combination of three heuristics that the combination of largest degree and largest enrolment can produce better results. Table 3.9 shows that the combination of LD-SD-LE and LD-LE-LWD obtain three and four best results respectively, while the other two combinations, LD-SD-LWD and SD-LE-LWD, each achieves two best results within the combination of three heuristics. The combination of all heuristics appear to work well but it still cannot obtain any best results when compared with other combinations. It is clear that the combinations of multiple heuristics in the AHO approach cannot create any feasible solutions to Exam_4.

Table 3.10 compares the best results of different approaches of the ITC2007 benchmark datasets with the best results obtained by the AHO approach. It can be noted that the approaches from [1] to [5] are the results from the competition, while the remaining approaches are obtained after the competition. It is observed that no best result has been achieved so far within the same computational time and that the obtained results are not close to other best approaches except for Exam_6, Exam_10 and Exam_12. Nevertheless, some of the results are better than Müller (2008) for Exam_10, Atsuta et al. (2008) for Exam_2 and Exam_6, Pillay (2008) for Exam_4, Exam_6, Exam_7 and Exam_10 and Burke et al. (2010f) for Exam_4, Exam_6 and Exam_8. It can also be noted that the AHO approach is simply a constructive one and that improvement is required in order to increase its solution quality. This is therefore considered to be an indirect comparison, because the other approaches are improvement strategies. The implementation of the improvement approach is discussed in Chapter 6.

Table 3.8: Comparison of basic AHO and AHO with shuffling best ordering for four different graph colouring heuristics for the ITC2007 benchmark datasets (LD = largest degree, LE = largest enrolment, LWD = largest weighted degree, SD = saturation degree, *inf.* = infeasible)

	Basic AHO				AHO - shuffling best ordering			
Problem	LD	LE	LWD	SD	LD	LE	LWD	SD
Exam_1	11155	12261	11897	11228	**11019**	12163	11950	11173
Exam_2	3272	3175	**3128**	3159	3164	3152	3212	3148
Exam_3	**19661**	20957	21204	19705	19920	21025	21077	19707
Exam_4	**23978**	*inf.*	*inf.*	24195	27399	*inf.*	*inf.*	31676
Exam_5	8033	8768	8678	**8032**	8253	8626	8655	8041
Exam_6	**28295**	58445	28425	28580	28595	28460	28625	28600
Exam_7	16002	**15573**	16115	16163	16032	16038	16001	16275
Exam_8	19802	20322	20124	20146	19921	20095	**19684**	20238
Exam_9	**2192**	2207	2248	2271	2212	2230	2207	2225
Exam_10	16828	17429	17301	17336	**16741**	17160	16847	17157
Exam_11	**44921**	48807	49482	46583	46695	47063	48288	47213
Exam_12	6502	6654	6755	**5858**	6430	6551	6189	6534

3.3.3 Discussion

The overall results of the proposed constructive approaches showed that the approach is very competitive when compared with other constructive approaches within the Toronto benchmark instances. Moreover, the approach also showed promising results for certain instances when compared with other improvement approaches within the Toronto and ITC2007 benchmark instances. As can be seen in the table of results (Table 3.4, 3.5, 3.6 and 3.10), the shuffling current best ordering and also alternating the heuristics, also contributed to better ordering. The results also demonstrated that the size of block or top-window of shuffling strategy should not be too large and the statistical analysis on the size of the shuffling strategy is shown later in this subsection.

Figures 3.3 to 3.10 demonstrate the contribution of the number of violated examinations at each iteration for four different graph colouring heuristics from one problem instance of the Toronto and ITC2007 benchmark datasets. Each problem instance illustrates the number of violated examinations for four different heuristic modifiers: custom, additive, multiplicative and exponential, in different figures. The illustrations are from the test of basic AHO with top-window size 5 and at 1000 iterations. Figures 3.3 to 3.6 show the pattern of largest degree, largest enrolment, largest weighted degree and saturation degree for yor83 I of the Toronto benchmark datasets.

As Figure 3.3 illustrates, there are significant differences in the number of violated examinations among all of these graph colouring heuristics. In Figure 3.3 (a), the number of

Table 3.9: Comparison of different heuristics strategies of AHO of the ITC2007 benchmark datasets (The bold entries indicate the best results for given heuristic combination type, while those in bold and italic indicate the best results found for the given problem) (LD = largest degree, SD = saturation degree, LE = largest enrolment, LWD = largest weighted degree, *inf.* = infeasible)

	Heuristic combinations										
	Two heuristics						Three heuristics				All heuristics
Problem	LD-SD	LD-LE	LD-LWD	SD-LE	SD-LWD	LE-LWD	LD-SD-LE	LD-SD-LWD	LD-LE-LWD	SD-LE-SLWD	LD-SD-LE-LWD
Exam_1	11217	11234	**11051**	11233	11231	12026	11123	11274	11311	11426	11175
Exam_2	3232	3042	3161	3156	3202	3137	3275	3182	**2880**	3253	3100
Exam_3	20139	**19288**	19948	20186	20373	21300	20225	20311	20151	20587	20317
Exam_4	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*
Exam_5	**8091**	8387	8411	8405	8307	8750	8295	8265	8713	8329	8248
Exam_6	28610	28735	28620	28690	**28595**	28650	28720	28755	28715	28770	28715
Exam_7	16100	16037	16156	16326	16177	18188	**15895**	16098	16136	16197	16223
Exam_8	19853	**19772**	20157	20335	20194	20415	20075	20271	19989	19900	20259
Exam_9	2198	2204	**2157**	2194	2201	2184	2254	2204	2214	2216	2238
Exam_10	17244	17016	17063	17244	17260	**16943**	17157	17194	17019	17239	17061
Exam_11	46134	47850	46160	48662	48841	49634	**45674**	47156	49261	50162	48969
Exam_12	8169	**6566**	7181	7444	8291	6587	8716	8620	8469	7456	9037

Table 3.10: Comparison of AHO with different approaches of the ITC2007 benchmark datasets

Problem	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	AHO
Exam_1	4370	5905	8006	6670	12035	4370	4633	8559	6235	4775	**4368**	11019
Exam_2	400	1008	3470	623	3074	385	405	830	2974	**385**	390	2880
Exam_3	10049	13862	18622	-	15917	9378	9064	11576	15832	**8996**	9830	19288
Exam_4	18141	18674	22559	-	23582	**15368**	15663	21901	35106	16204	17251	23978
Exam_5	2988	4139	4714	3847	6860	2988	3042	3969	4873	**2929**	3022	8032
Exam_6	26950	27640	29155	27815	32250	26365	25880	28340	31756	**25740**	25995	28295
Exam_7	4213	6683	10473	5420	17666	4138	**4037**	8167	11562	4087	4067	15573
Exam_8	7861	10521	14317	-	16184	7516	**7461**	12658	20994	7777	7519	19684
Exam_9	1047	1159	1737	1288	2055	1014	1071	-	-	**964**	-	2157
Exam_10	16682	-	15085	14778	17724	14555	14374	-	-	**13203**	-	16741
Exam_11	34129	43888	-	-	40535	31425	29180	-	-	**28704**	-	44921
Exam_12	5535	-	5264	-	6310	5357	5693	-	-	**5197**	-	5858

[1]-Müller (2008); [2]-Gogos et al. (2008); [3]-Atsuta et al. (2008); [4]-De Smet (2008); [5]-Pillay (2008); [6]-Müller (2009); [7]-McCollum et al. (2009); [8]-Gogos et al. (2009); [9]-Pillay (2010a); [10]-Burke et al. (2010f); [11]-Gogos et al. (2010a); [12]-Turabieh and Abdullah (2011)

violated examinations generated by the saturation degree heuristic is very low compared with other heuristics, especially at the end of the iteration. Thus, the saturation degree heuristic has greater chances of creating feasible solutions. On the other hand, among the static types of heuristic, the largest degree heuristic generates a less number of violated examinations, while the largest enrolment and the largest weighted degree heuristic contribute a higher number of violated examinations. Figure 3.3 (a) demonstrates the largest weighted degree contributes the highest number of violated examinations. Nevertheless, these heuristics still fail to generate a feasible solution throughout the iteration, except for the saturation degree heuristic. The yor83 I of the Toronto benchmark dataset is one of the difficult problem instances to be solved with conflict density 0.29. This might be the reason why the static types of heuristics fail to generate a feasible solution when implemented with a different type of heuristic modifier (see Figures 3.4 to 3.6). It is also shown in Figure 3.3 that that there is no convergence to each heuristic over iterations. This is may be due to the employment of custom type of heuristic modifier which only uses heuristic value for the difficulty value, and it requires no increment to the difficulty value if an examination cannot be scheduled to a time-slot.

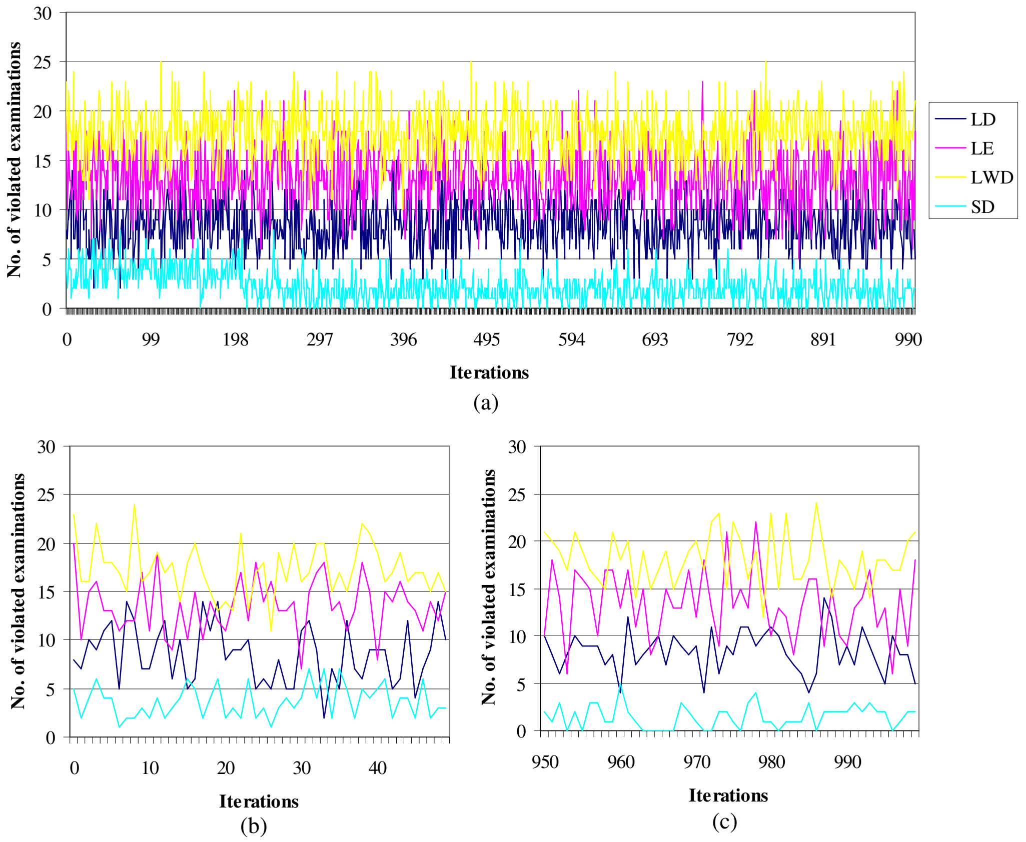
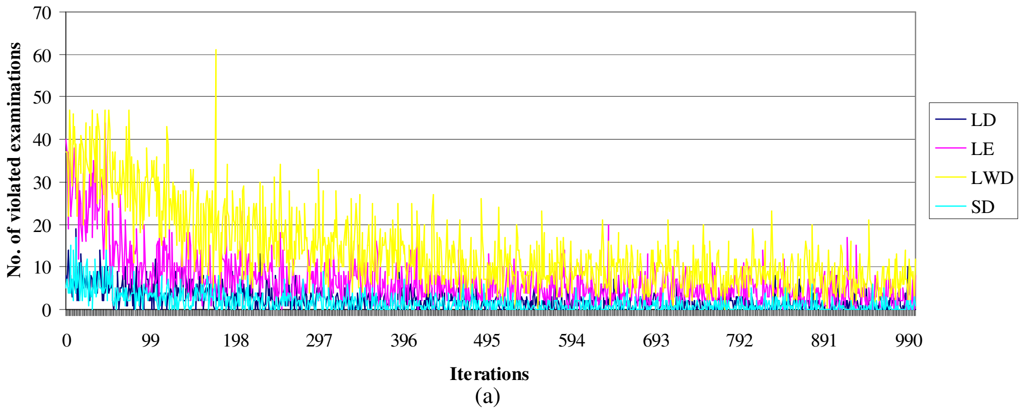


FIGURE 3.3: Number of violated examinations at each iteration for yor83 I of the Toronto benchmark datasets with custom (C) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations



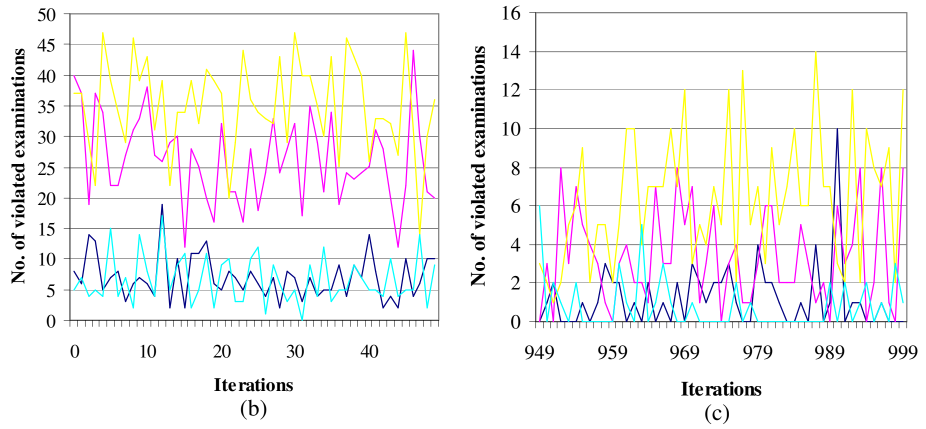


FIGURE 3.4: Number of violated examinations at each iteration for yor83 I of the Toronto benchmark datasets with additive (AD) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

Different patterns of behaviour can be seen in Figure 3.4 when the additive heuristic modifier is employed. Figure 3.4 (a) shows a downward movement in the number of violated examinations for all of the static types of heuristics, while the behaviour of the saturation degree heuristic remains the same throughout the iteration. Within the first fifty iterations (as shown in Figure 3.4 (b)), as the iteration starts, the behaviour of the number of violated examinations is about the same as in Figure 3.3 (b) when implemented with the custom heuristic modifier. The heuristic modifier keeps the information of the number of violated examinations to be interacted with. However, at the end of the iteration (as shown in Figure 3.4 (c)), some of the static heuristics have successfully generated feasible solutions.

The multiplicative heuristic modifier applied to yor83 I in Figure 3.5 shows the increment in the number of violated examinations compared with the additive heuristic modifier in Figure 3.4. Although it essentially shows a downward movement, at the end of the iteration (as shown in Figure 3.4 (c)), the number of violated examinations remains slightly higher for all heuristics compared with the additive heuristic modifier in Figure 3.4. With this heuristic modifier, as can be seen in Figure 3.5, only largest degree and

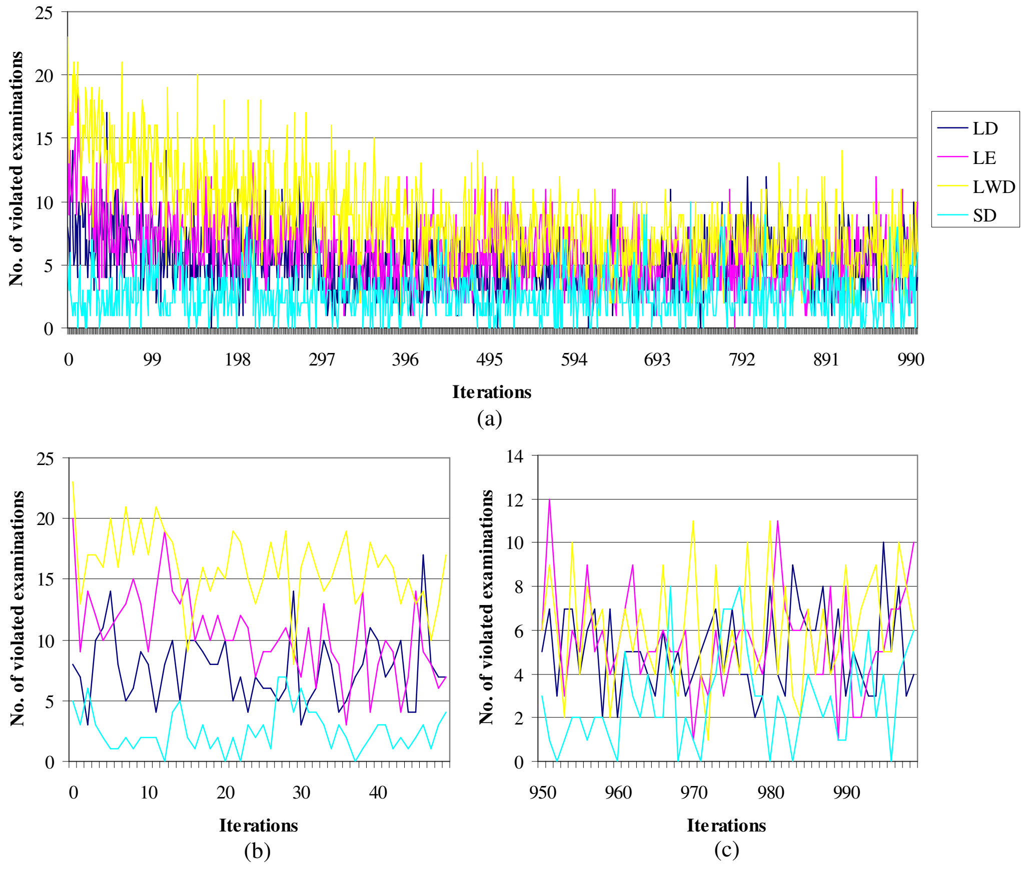


FIGURE 3.5: Number of violated examinations at each iteration for yor83 I of the Toronto benchmark datasets with multiplicative (MP) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

saturation degree can generate feasible solutions, but the frequency remains too small for the largest degree heuristic.

The exponential heuristic modifier in Figure 3.6 shows an obvious movement for the static heuristics. At the beginning of the iteration, the number of violated examinations is decreased gradually while the number of violated examinations for saturation degree heuristic is rising and after a while remained constant. After half way through the iteration, the number of violated examinations for the static type of heuristic suddenly grows very high and fails to reach a feasible solution. On the other hand, the number of violated examinations for the saturation degree heuristic remains constant and can still generate a feasible solution even though this is not as often as the custom, additive and multiplicative heuristic modifiers. From this observation, it is indicated that, for the yor83 I problem, the longer the run, the greater the chances of generating the higher number of violated examinations especially, for the static type of heuristic. It can be noted that the test is run with the exponential modifier type and it is assumed that increasing the difficulty in greater amounts of difficulty value for this problem

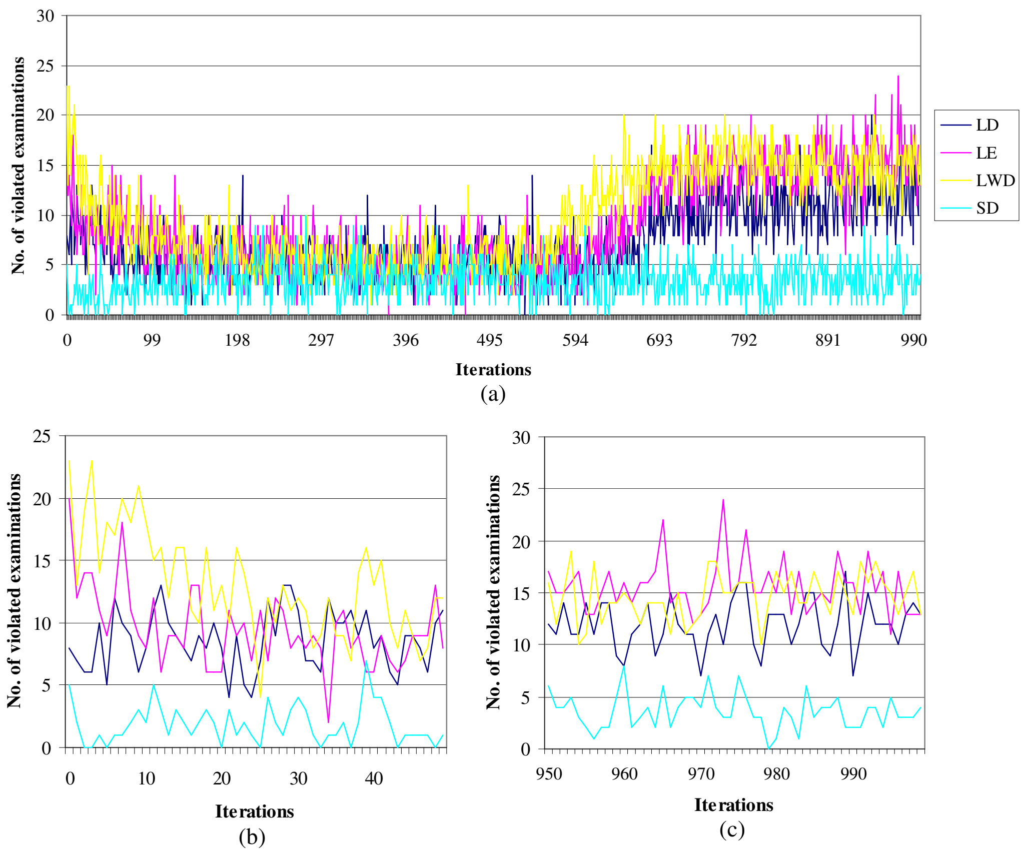


FIGURE 3.6: Number of violated examinations at each iteration for yor83 I of the Toronto benchmark datasets with exponential (EX) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

could drastically change the examination ordering and at the same time give a higher possibility of generating infeasible solutions. On the other hand, the success of the ordering with saturation degree heuristic and exponential heuristic modifier is due to the nature of this heuristic that can adapt the difficulty value at each assignment. This has given advantage for the saturation degree heuristic for better ordering and this feature is different from the static type of heuristics that always use the same heuristic value for ordering. General observation and the behaviour of other problem instances of Toronto benchmark datasets can be viewed in Appendix A.

A two-way analysis of variance (ANOVA) tested on different graph colouring heuristics and different modifier types used, i.e. custom, additive, multiplicative and exponential, was performed. From the statistical analysis, $F_{(15,9584)} = 19.514$ and $\rho(0.000) < 0.05$. The test revealed that there are significant differences between the solution quality for the exponential heuristic modifier and that of the other three modifier types. It has also been shown that there is a significant difference between the solution quality within heuristic types, with the largest degree heuristic producing significantly different results from those obtained with the largest enrolment, largest weighted degree and saturation degree heuristics. When comparing the effect on the block and top-window size, different solution qualities are tested with different sizes of block and top-window from {none, 2, 3, 4, 5, 6, 7, 8, 9}. In general, there is a significant difference of solution quality between the different sizes of block or top-window where $F_{(8,10215)} = 18.098$ and $\rho(0.000) < 0.05$.

Table 3.11 shows the effect between different size of block or top-window on solution quality. In the table, the solution quality without the shuffling strategy (none) and with shuffling strategy (All sizes) shows a significant difference. This indicates that the incorporation of shuffling strategy has helped the approach to obtain better solution quality. However, the solution quality obtained from the block or top-window sizes 2, 3, 4, 5, and 6 are not significantly different and this shows that shuffling with these sizes could obtain close solution quality.

Table 3.11: The effect of different size of block/top-window on the solution quality of the Toronto benchmark datasets ($\neq$ indicates significant differences among the sizes)

Size	**Effect**	**Size**
none	$\neq$	All sizes
2	$\neq$	none, 7, 8, 9
3	$\neq$	none, 7, 8, 9
4	$\neq$	none, 7, 8, 9
5	$\neq$	none, 8, 9
6	$\neq$	none, 9
7	$\neq$	none, 2, 3, 4
8	$\neq$	none, 2, 3, 4, 5
9	$\neq$	none, 2, 3, 4, 5, 6

Figures 3.7 to 3.10 show the behaviour of Exam_11 from the ITC2007 benchmark datasets, tested with the same parameter setting as yor83 I for custom, additive, multiplicative and exponential heuristic modifiers. From the figures, it is interesting to see that the behaviour of the saturation degree and largest degree heuristics is about the same in terms of the number of violated examinations generated during the timetable construction. Further, the number of infeasible solutions achieved for the largest degree and saturation degree heuristics are very similar, while the largest enrolment and the largest weighted degree heuristics are varied in certain ways. Within the custom heuristic modifier (as shown in Figure 3.7), the largest degree and saturation degree have constant movement throughout the iteration and sometimes generate feasible solutions. The largest enrolment and the largest weighted degree heuristic also have constant movement. However, the numbers of violated examinations for both heuristics are very high and far from achieving the feasible solution. This behaviour is about the same as for yor83 I when the approach utilised the custom type of heuristic modifier. The custom

heuristic modifier made no changes to the difficulty value due to that only the heuristic value is used to evaluate the difficulty of an examination.

As the additive heuristic modifier is employed with Exam_11 (see Figure 3.8), the number of violated examinations is gradually decreased for all heuristics. More feasible solutions are achieved for all heuristics. However, the number of infeasible solutions for the largest weighted degree is still high. The same behaviour is observed for Exam_11 when using the multiplicative heuristic modifier (see Figure 3.9). This time the largest weighted degree heuristic generates more improvement with a lower number of infeasible solutions. Moreover, by using this modifier the other heuristics also generate feasible solutions more often compared with the custom and additive heuristic modifiers.

Using the exponential heuristic modifier shows more improvement in generating feasible solutions. As illustrated in Figure 3.10, although all heuristics generate a very high number of violated examinations at the beginning, in a short time they can generate feasible solutions for the remaining iteration. Figure 3.10 (c) is an illustration of the behaviour of all heuristics. It can be observed that the largest enrolment has a higher number of violated examinations at the end of the iteration but still managed to achieve feasibility. It can also be noted that the conflict density of Exam_11 is 0.0262, illustrating the difficulty of the problem in terms of examinations in conflict. However, although Exam_11 is less difficult than yor83 I in terms of conflict density value, it is difficult in satisfying the room capacity as well as the other time-slots and the room hard constraints requirement. Behaviour on different problem instances of ITC2007 benchmark datasets can be viewed in Appendix A.

From the two-way analysis of ANOVA of the ITC2007 benchmark datasets, it has been shown that there is a significant difference between the solution quality obtained with the largest degree heuristic compared with the other three heuristics, and also that there is a significant difference between the solution quality of the multiplicative heuristic modifier with the other three heuristic modifiers, where $F_{(15,10384)} = 409.052$ and $\rho(0.000) < 0.05$. It is also demonstrated in other test that there is a significant difference in the solution quality when implemented with different block/top-window sizes with $F_{(8,9391)} = 104.930$ and $\rho(0.000) < 0.05$.

The AHO approach for constructing examination timetables draws upon the work by Burke and Newall (2004). Recent investigation on AHO approach shows a significant improvement to the solution quality for certain problem instances when compared with the study by Burke and Newall (2004). In Burke and Newall (2004), two types of adaptation i.e. hard constraint only and hard and soft constraints were considered for constructing examination timetables. However, when using the approach condering hard constraint only, the approach increased the heuristic modifier whenever an examination

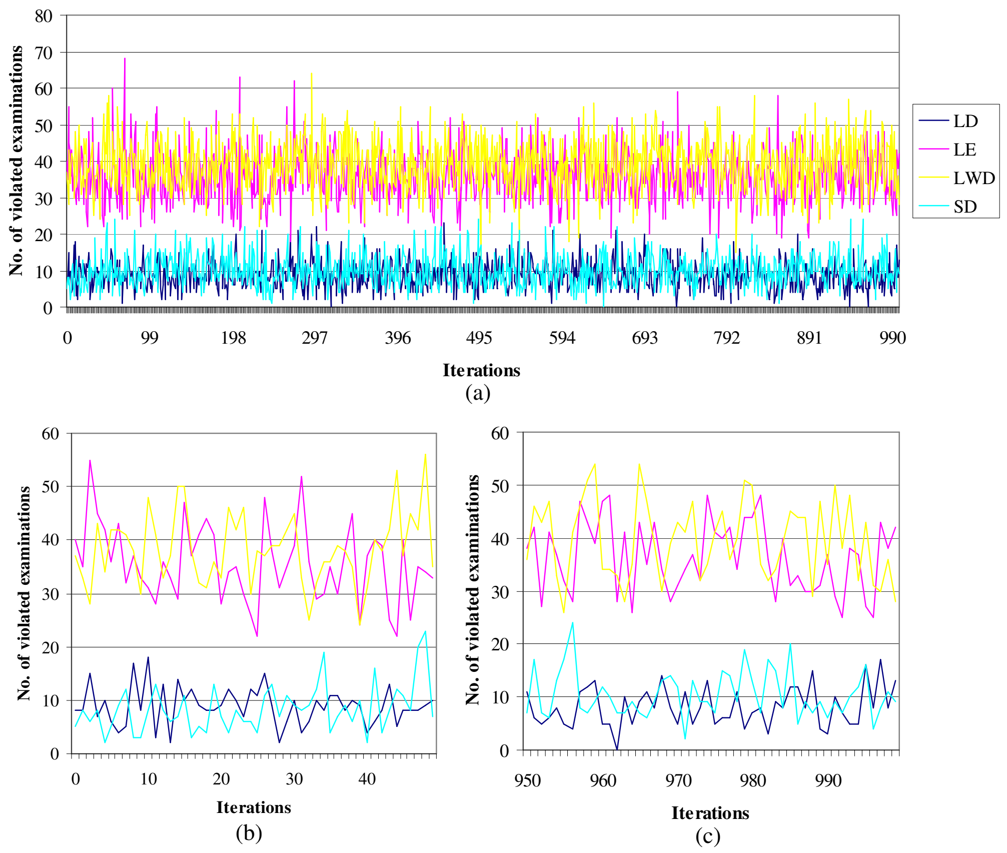


Figure 3.7: Number of violated examinations at each iteration for Exam_11 of the ITC2007 benchmark datasets with custom (C) heuristic modifier of tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

cannot be assigned to a timetable and the run was terminated whenever a feasible solution was found. In the second type of adaptation, the soft constrains were optimised considering examination cost that is less than the maxCost value and this maxCost value was identified based on the highest cost found for examination assignment during a dummy run. Moreover, the approach only considered exponent heuristic modifier and a random value in order to increase the difficulty value. The approach also considered the 'top-window' strategy.

Our approach follows the same concept as in the first approach of Burke and Newall (2004) where the heuristic modifier is increased whenever an examination cannot be scheduled into a timetable. However during the timetabling process, our approach considered the best time-slots assignment in order to reduce the examination cost throughout the iterations. Our approach considered top-window strategy for the dynamic type of heuristic and block strategy specialised for the static type of heuristics. Moreover, different sizes of shuffling strategies were tested in order to find the best size combination statistically. We tested with different type of heuristic modifiers, i.e. custom, additive, multiplicative and exponential, in order to differentiate their performance in terms of

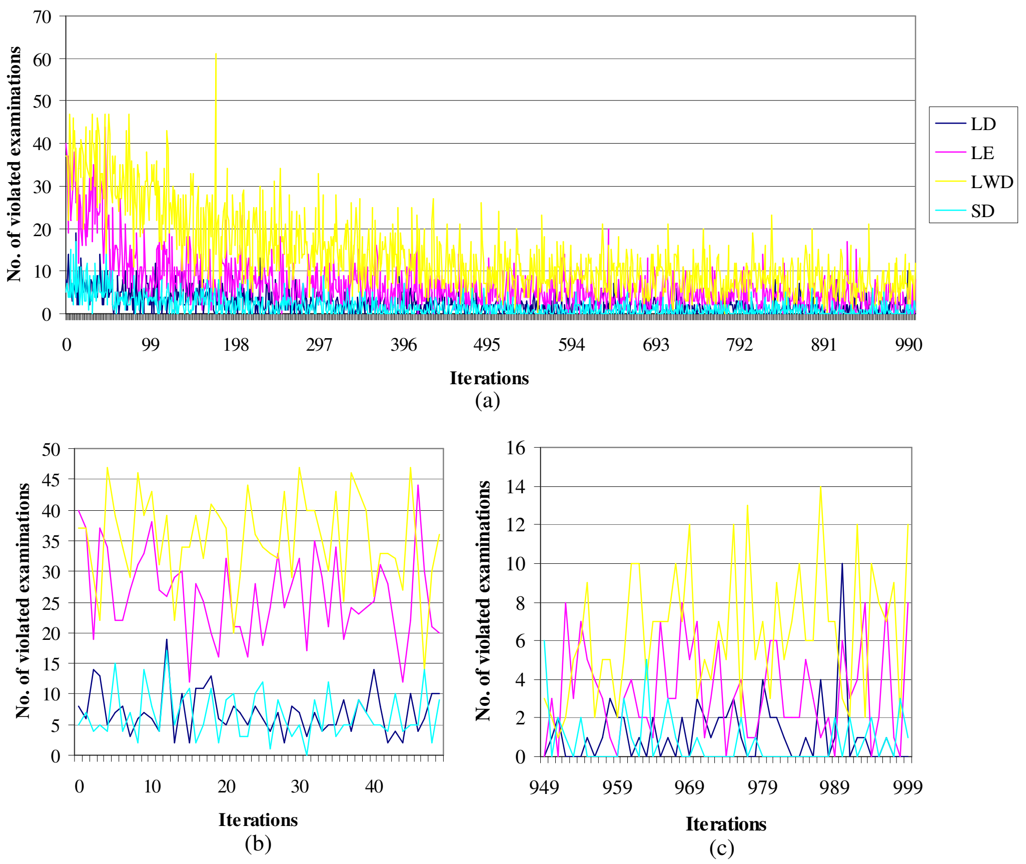


FIGURE 3.8: Number of violated examinations at each iteration for Exam_11 of the ITC2007 benchmark datasets with additive (AD) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

solution quality and the number of examinations that cannot be scheduled obtained from the timetabling process. Further, we considered shuffling the best examination ordering whenever there is no improvement to the solution quality for a certain time and at the same time combined a number of graph colouring heuristics in the AHO framework.

The results have shown that our approach is comparable with the approach by Burke and Newall (2004) where for half of the instances of the Toronto benchmark dataset, we obtained better results. Moreover, Burke and Newall (2004) did not provide results for some problem instances i.e. pur93 I and rye92. On the other hand, results for the remaining instances are closed to those of Burke and Newall (2004). However, our approach takes longer processing time when compared with Burke and Newall (2004). This is due to that our approach incorporates a strategy to find the least penalty time-slot that considered all time-slots to be evaluated and the incorporation of the shuffling strategy also contribute to the increment of the running time.

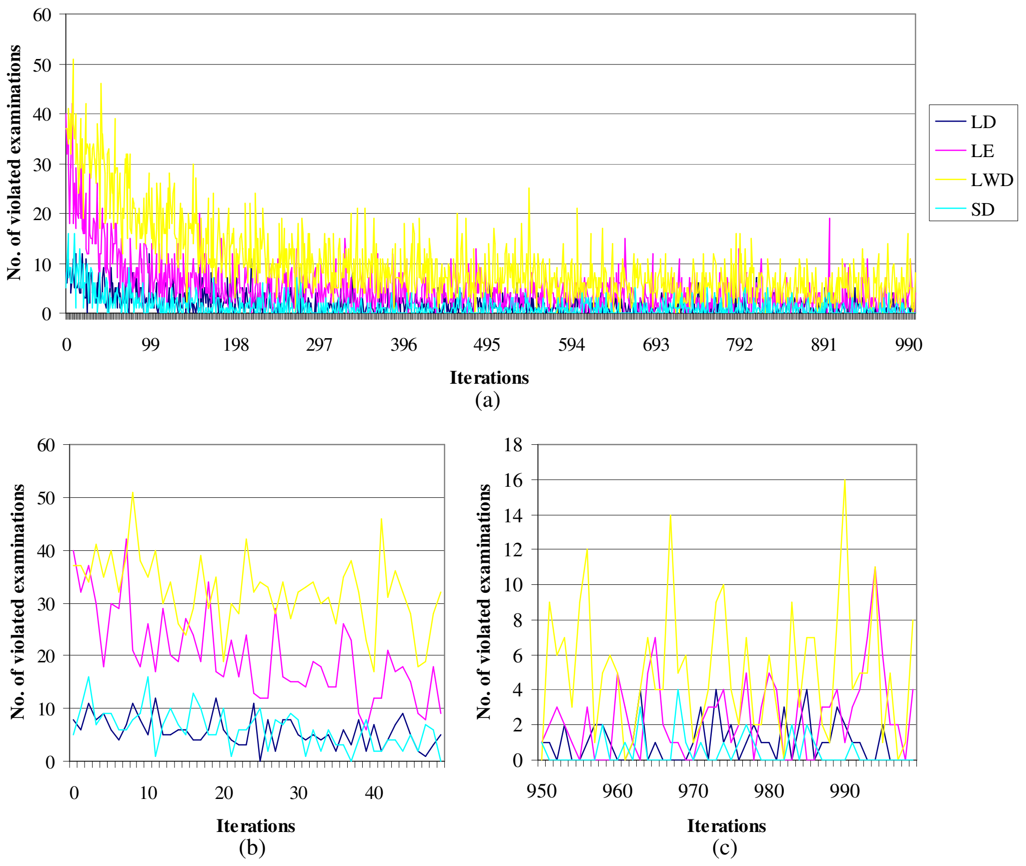


FIGURE 3.9: Number of violated examinations at each iteration for Exam_11 of the ITC2007 benchmark datasets with multiplicative (MP) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

3.4 Conclusion

This chapter investigated the adaptive heuristic ordering that schedules examinations within a constructive approach drawing on the work of Burke and Newall (2004) that incorporated a heuristic modifier to order examinations based on difficulty. The approach explored different graph colouring heuristics with different combinations of heuristic modifier and block and top-window size. Further, in the present study the current best ordering was shuffled using the top-window strategy in order to obtain better ordering and, by alternating the heuristics in the list, the choice of the difficult examinations can be varied. A stochastic component was incorporated into the process of assigning a selected examination to a time-slot. The AHO approaches can produce solutions comparable to the other approaches. The difficulty levels generated by combining a graph coloring heuristic and a heuristic modifier were used in ordering the examinations for the timetabling process. From this observation, good approximate solutions can be obtained

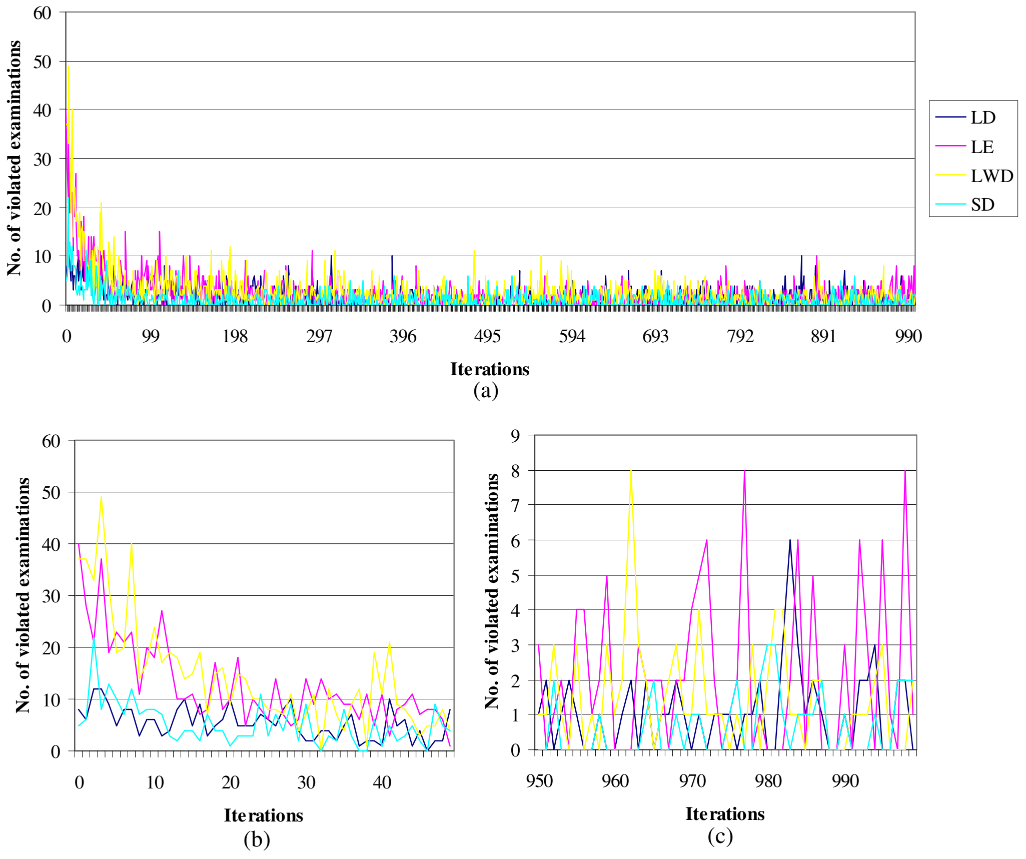


Figure 3.10: Number of violated examinations at each iteration for Exam_11 of the ITC2007 benchmark datasets with exponential (EX) heuristic modifier tested with basic AHO with top-window size five (a) the whole picture of each heuristic behaviour; (b) the behaviour of the first fifty iterations; and (c) the behaviour of the last fifty iterations

by increasing the difficulty in certain ways. As a dynamic graph coloring heuristic, saturation degree has produced the largest number of the best results compared with other static heuristics applied to the Toronto benchmark datasets, while the ITC2007 benchmark datasets performed differently. In considering the appropriate heuristic modifier, the exponential approach is the best for largest degree and saturation degree for the Toronto benchmark datasets, while the largest enrolment and the largest weighted degree are varied. The performance of the algorithm on the ITC2007 benchmark datasets also varies with different types of heuristic modifiers. In comparing the performance of different heuristics, it is statistically validated that the saturation degree performed better than the other graph colouring heuristics in terms of producing good quality feasible solutions. Furthermore, the statistical analysis showed that different graph colouring heuristics and different heuristic modifiers could affect the solution quality. The block and top-window size approaches in this study have varied since the incorporation of a stochastic element in the AHO approach. The statistical test demonstrated that there was a significant difference when employing different block or top-window sizes to the AHO approach. It was shown that, by alternating the graph colouring heuristic, we could achieve better ordering and at the same time improve the solution quality. Nevertheless, this approach is simple, effective and requires less computational time. Hence, it has potential for practical use.

In the next chapter, the approach is further enhanced by combining graph colouring heuristics with a heuristic modifier in a linear form, and the incorporation of a weight value for each parameter has been shown to assist in changing the examination ordering based on its difficulty.

Chapter 4

Adaptive Linear Combination of Heuristic Orderings in Constructing Examination Timetables

This chapter presents an adaptive linear combination of heuristic orderings in constructing examination timetables that are based upon the previous work presented in Chapter 3. The aim is to obtain new difficulty values that are extracted from the combination of graph colouring heuristics with a heuristic modifier using a linear approach. Different weights are assigned to each parameter and the effectiveness of the proposed approach is analysed. The effect of weights associated with ordering on the quality of the examination schedules is explored using different graph colouring heuristics. Further, the chapter investigates the effectiveness of weight changes that can change automatically during the examination assignment to suit different problem instances. An explanation of the adaptive linear combination of heuristic ordering and the strategies of weight changes is provided in Section 4.1. Section 4.2 describes the implementation and analysis of the results on two benchmark datasets. Finally, conclusions are drawn in Section 4.3.

4.1 An Adaptive Linear Combination of Heuristics Orderings

With most of the approaches taken within the overall family of constructive methods, it is often the case that a single heuristic is used during the initial ordering phase. In considering the difficulty of an examination, it is useful to take into account other factors that affect the ordering of examinations. Considering many factors at one time represents the real world situation. The difficulty of scheduling an examination can be approximated more reliably if several heuristics lend support to the final ordering of examinations. Consequently, the current constructive study by Burke et al. (2010e) combined graph colouring heuristics with weights within a linear approach as to measure the difficulty of a vertex of weighted graph. The study used the vertex-selection heuristics to represent the difficulty of a vertex and it was continually updated throughout the timetabling process.

Studies by Johnson (1990) and Asmuni et al. (2009) have also deployed this strategy by considering more than one heuristic at one time and it has been shown that it has an effect on the ordering of the examinations. Based on the '*difficulty factor*', Johnson (1990) used graph colouring heuristics, i.e. the combination of largest enrolment and largest degree as an ordering strategy for assigning examinations to time slots. Several variations of relative weight of each criterion were considered in order to produce a number of different feasible timetables. Further, Asmuni et al. (2009) combined two graph colouring heuristics within the framework of a fuzzy methodology in order to deal with uncertainty in ordering the examinations based on its difficulties. Three graph colouring heuristics were used, i.e. largest degree, largest enrolment and saturation degree with three combinations of two heuristics. The studies indicated that the solution quality was superior compared with using only a single heuristic. Encouraged by these studies, the present investigation on the implementation of the heuristic modifier (described in Chapter 3) is extended by combining graph colouring heuristics with a heuristic modifier using a linear approach.

An adaptive linear combination of heuristic orderings in this study is a combination of a number of normalised graph colouring heuristics with normalised difficulty measures of the *heuristic modifier* where the heuristic modifier was introduced by Burke and Newall (2004). Adaptive linear combination of heuristic orderings is a flexible approach because different weights can be assigned to different parameters used in the combination. Information from the chosen heuristics and heuristic modifiers are used to identify new orderings of examination to be scheduled. The new ordering of an examination based on an adaptive linear combination of heuristic orderings is represented by the following equation:

$$difficulty\_score_i(t) = \sum_{j=1}^{n} w_j \times heuristicN_{ij} + w_{HM} \times heurmod_i(t) \tag{4.1}$$

where,

$$heuristicN_{ij} = \frac{heuristic_{ij}}{maxheuristic_j} \tag{4.2}$$

$$heurmodN_i(t) = \frac{heurmod_i(t)}{choosemax(heurmod(t))} \tag{4.3}$$

$$\sum_{j=1}^{n} w_j + w_{HM} = 1 \tag{4.4}$$

The $difficulty\_score_i(t)$ is used as a difficulty measure for examination i at iteration t based on the information evaluated. In the present study, a zero-one normalisation method is used to obtain the normalised value between 0 and 1 for for each $heuristicN_{ij}$ and $heurmodN_i(t)$ to ensure a simple generalisation characteristic of the problem data. The $heuristicN_{ij}$ in equation (4.2) is the normalised graph colouring heuristic j for examination i while $heurmodN_i(t)$ in equation (4.3) is the normalised heuristic modifier for examination i at iteration t. The $maxheuristic_j$ is the maximum identified value of heuristic j while the $choosemax$ function is provided to give an alternative to the heuristic modifier to change dynamically or statically. Given in equation (4.4) the total weight of heuristic j, w_j and heuristic modifier, w_{HM} is equal to 1.

Two types of graph colouring heuristics were used in this suite of experiments. In order to compare their contribution to solution quality, a series of experiments has been carried out, firstly, using each single heuristic separately, and subsequently combining both heuristics with and without a heuristic modifier. The purpose was to compare the performances of single and multiple heuristics and to identify the most effective combination of heuristics with the heuristic modifier. It should be noted that, although not investigated here, more graph colouring heuristics can be used within this approach.

In our previous study, Abdul Rahman et al. (2009) used various *modify* function for heuristic modifiers to change the order of examinations based on their difficulty value. The difficulty values were updated and increased with four strategies: custom, additive, multiplicative and exponential. The examination ordering was based on only one graph colouring heuristic during the timetabling. In this study, the same modify function i.e.

additive and exponential are employed and the linear approach adapts the normalisation strategy in order to generalise the ordering of $difficulty\_score$ by combining a number of graph colouring heuristic with a heuristic modifier. Section 3.1.2 in Chapter 3 describes the type of modify function of the heuristic modifier.

Once the heuristic modifier and the difficulty of an examination have been updated, the difficulty value of the heuristic modifier is normalised statically or dynamically based on the *choosemax* function. After all the heuristic values and the heuristic modifier have been updated with the chosen weights, all the values are summed up to obtain $difficulty\_score$. The examinations are then ordered decreasingly based on $difficulty\_score$ before an assignment is made. The pseudocode of the implemented approach in this thesis is described in Algorithm 8.

Algorithm 8 Construction of a timetable based on adaptive linear combination of heuristic orderings

```
for $t = 1$ to number of iterations do
  for $i = 1$ to number of examinations do
    for $j = i$ to number of examinations do
      Calculate the normalise value of chosen heuristics: $choosemax(heurmodN_j(t))$
      , $HeuristicN_{LD,j}$, $HeuristicN_{SD,j}$ (refer equation 4.2, 4.3 and 4.4) with weight
      value
      Calculate the $difficulty\_score_j(t)$ for each examination according to the cho-
      sen heuristics using equation 4.1
    end for
    Sort($difficulty\_score(t)$) in a decreasing order
    if $i$ can be scheduled then
      Schedule $i$ in the time-slot with the least penalty
      In the case of the availability of multiple time-slots with the same penalty,
      choose one randomly
    else
      Increase and modify heuristic modifier of $i$ (refer Subsection 3.1.2)
    end if
    if Saturation_degree then
      Update the Saturation_degree
    end if
  end for
  Evaluate solution, store if it is the best found so far
end for
```

Tables 4.1 and 4.2 show an example of how the ordering is achieved using various combinations of heuristics after a certain number of iterations. In this example, it is assumed that the total number of time-slots is 10. Table 4.1 illustrates the ordering using a single heuristic. Since we want to use only one heuristic for the ordering, then the weight value for the single heuristic that is chosen is set to 1.0 and the other heuristics are set as 0. Referring to column 2 of an unordered list in Table 4.1, we assumed all the largest degree values. In that case, the *maxheuristic* for largest degree is equal to 19. The calculation of the *difficulty_score* value for the single ordering of LD in column 5 is based on equation 4.1 where the difficulty score for e4 = (1.0)19/19 = 1.0, e1 = (1.0)17/19 = 0.89 and so on.

TABLE 4.1: Examples of ordering by combinations of single heuristics (LD = largest degree; SD = saturation degree; HM = heuristic modifier; diff_score = difficulty_score)

Unordered list			Ordering by single LD		Ordering by single SD		Ordering by single HM	
exams	LD	HM	exams	diff_score	exams	diff_score	exams	diff_score
e1	17	4	e4	1.00	e2	-	e2	1.00
e2	14	20	e1	0.89	e4	0.1	e4	0.75
e3	16	10	e10	0.84	e6	0.1	e6	0.70
e4	19	15	e3	0.84	e10	0.1	e8	0.60
e5	9	0	e2	0.74	e3	0.0	e10	0.60
e6	11	14	e6	0.58	e1	0.0	e3	0.50
e7	8	7	e5	0.47	e9	0.0	e7	0.35
e8	8	12	e7	0.42	e5	0.0	e1	0.20
e9	8	0	e9	0.42	e7	0.0	e9	0.00
e10	16	12	e8	0.42	e8	0.0	e5	0.00

The example for SD is shown in column 2. The single ordering for SD is dynamic. After each assignment of a time-slot, the new examination ordering is obtained. Initially, as implemented by Abdul Rahman et al. (2009), the saturation degree value is set to 0. Assumed that e2 is chosen as the first examination to be assigned to a time-slot. Once e2 is assigned to a time-slot, the saturation degree value for the unscheduled examinations is updated by considering the conflict with other examinations in a previous assignment. Assumed that e4, e6 and e10 have conflicts with e2, then the saturation degree of these examinations are increased by one and the *difficulty_score* (using equation 4.1) for e4 = (1.0)1/10 = 0.1, e6 = (1.0)1/10 = 0.1 and e10 = (1.0)1/10 = 0.1 while there rest are zero due to no conflict with e2. The calculation of the difficulty value for each unassigned examinations continues until no more examinations are to be assigned to a time-slot. The ordering by single heuristic modifier (HM) in column 3 in Table 4.1 is based on the number of times an examination cannot be scheduled during the previous iterations. It is assumed that the figures in column 3 are the number of times these examinations cannot be assigned into a timetable during the previous iterations. By considering equation 4.1, the *difficulty_score* for e2 = (1.0)20/20= 1.0, e4 = (1.0)15/20 = 0.75, e6 = (1.0)14/20 = 0.70 and so on.

Table 4.2 illustrates the example combination of more than one heuristic. It is assumed that the total number of time-slot to be assigned is 10. The ordering by LDSD is a dynamic ordering. Considering the weight for LD, $w_{LD} = 0.2$ and the weight for SD,

TABLE 4.2: Examples of ordering by combinations of multiple heuristics (LD = largest degree; SD = saturation degree; HM = heuristic modifier; diff_score = difficulty_score)

Ordering by LDSD		Ordering by LDHM		Ordering by LDSDHM	
exams	diff_score	exams	diff_score	exams	diff_score
e2	-	e4	0.800	e2	-
e4	0.280	e6	0.676	e4	0.500
e1	0.200	e10	0.648	e10	0.408
e3	0.100	e3	0.568	e6	0.396
e10	0.100	e8	0.564	e3	0.368
e6	0.100	e7	0.364	e8	0.324
e5	0.000	e1	0.279	e1	0.299
e7	0.080	e2	0.147	e7	0.224
e9	0.000	e5	0.095	e5	0.095
e8	0.000	e9	0.084	e9	0.084

w_{SD} = 0.8. By using equation (4.2), the *difficulty_score* for this combination for e4 = (0.2)(19/19) + (0.8)(0.1) = 0.28, for e1 = (0.2)(17/19) + (0.8)(0.2) = 0.34 and so on where it is based on the combination of information from the largest degree and saturation degree heuristics. Furthermore, it is assumed that e2 is the first examination to be chosen for assignment and it has been assigned to a time-slot and assuming also that e2 has conflict only with e4 and e1. In this case, the saturation degree values for e4 and e1 are increased by 1. By using equation 4.1 and considering the largest degree value from Table 4.1, the *difficulty_score* for this combination for e4 = (0.2)(19/19) + (0.8)(1/10) = 0.280, for e1 = (0.2)(17/19) + (0.8)(1/10) = 0.258, e3 = (0.2)(16/19) + (0.8)(0/10) = 0.168 and so on, where these calculations are based on the combination of information from the largest degree and saturation degree heuristics. Let us consider the weight for LD, wLD = 0.2 and the weight for HM, wHM = 0.8 for ordering the examinations using combination of LDHM. Considering the largest degree and HM values from Table 4.1, the *difficulty_score* (equation 4.1) for e4 = (0.2)(19/19) + (0.8)(15/20) = 0.800, e6 = (0.2)(11/19) + (0.8)(14/20) = 0.676 , e10 = (0.2)(16/19) + (0.8)(12/20) = 0.648 etc. In the next combination of heuristics, let us consider the weight for LD, wLD = 0.2, the weight for SD, wSD = 0.4 and the weight HM, wH M = 0.4 for ordering the examinations with combination of LDSDHM. It is assumed that e2 is the first examination to be chosen for assignment at certain iteration and it has been assigned to a time-slot and has conflict only with e4 and e1. Considering the information from Table 4.1, the *difficulty_score* for e = (0.2)(19/19) + (0.4)(1/10) + (0.4)(15/20) = 0.500, e10 = (0.2)(16/19) + (0.4)(0/10) + (0.4)((12/20) = 0.408, e6 = (0.2)(11/19) + (0.4)(0/10) + (0.4)(14/20) = 0.396 etc.

4.1.1 The *choosemax* Function

The normalised value of the heuristic modifier is determined by the *choosemax* function that gives significant modification to the $heurmodN_i(t)$:

- *Static (S).* The $heurmod_i(t)$ is normalised with the total number of iterations used in the algorithm. The larger the $heurmod_i(t)$, the more significant the value of $heurmodN_i(t)$.
- *Dynamic (D).* The $heurmod_i(t)$ is normalised with the current maximum number of *heurmod* of all examinations that change during the iteration. This value continues to change until the end of the iteration.

4.1.2 The Weight Assignment

Since this approach required weight assignment for each parameter, this study needs a strategy to assign the weight value. Each of the heuristics and the heuristic modifiers is assigned with different weight values. Using this approach, the weight values are assigned to each heuristic and heuristic modifier with the value from 0 to 1 with a 0.1 increment for each variable. The total of all weight values is equal to 1 (equation (4.4)). The combination of these weight values is tested for each of the variables in order to assess the performance of the heuristics and the heuristic modifier when different weight values are incorporated. It is important to know which heuristic is performing well and to note the importance of the heuristic modifier in this combination, so that the higher weight value is given to the appropriate parameters.

4.1.3 Shuffling the Ordering of Examinations

The present study employed the shuffling strategy used in our previous study (Abdul Rahman et al., 2009) in order to shuffle the examinations in the ordering, where the *top-window* (TW) strategy is adapted to choose examinations. These are ordered based on the *difficulty_score* and from a fixed size of top-window, an examination is chosen randomly. The insight of this strategy is to give more possibility to an examination to be chosen from a group of difficult examinations. An appropriate examination to be chosen might appear in a certain size of grouped examinations that has been ordered based on the *difficulty_score*. The initial test has shown that the incorporation of the shuffling strategy could assist in finding a better examinations ordering. This study uses the top-window size from two to nine, as suggested by Abdul Rahman et al. (2009). Since there is also a possibility that examinations have the same value of difficulty score, another strategy introduces a *random* preference (REQ) in order to choose different examinations when several sequences of examinations have equal scores.

4.1.4 Strategies of Weight Changes

This section introduces three different strategies to change the weights of each heuristic and heuristic modifier during the timetable construction. As reported in Chapter 3, the heuristics are changed during the timetable construction by alternating the graph colouring heuristics in the list. The alternation of graph colouring heuristics is based on the improvement to the solution quality. Due to the different number of violated examinations detected during examination assignment when using different heuristics, it is advantageous to change dynamically the examination weights during the examinations assignment. In this case, the number of violated examinations could be varied and different examination ordering could be obtained when a different heuristic modifier is incorporated. To simplify implementation, the weight combinations are divided into four main groups related to the contribution of current heuristics, namely, highLD, highSD, highHM and balance. The weight combinations of each group are illustrated in Table 4.3 and there are determined adhocly. The three strategies introduced in this section adaptively change the weight values for each heuristic and are discussed in the following subsection. It can be noted that these strategies are employed only with the combination of LDSDHM tested with the dynamic heuristic modifier and top-window approach.

Table 4.3: The grouping of different weight combinations for LDSDHM

Group	Weight Combination
highLD	(0.8, 0.1, 0.1), (0.7, 0.2, 0.1), (0.7, 0.1, 0.2)
highSD	(0.1, 0.8, 0.1), (0.1, 0.7, 0.2), (0.2, 0.7, 0.1)
highHM	(0.1, 0.1, 0.8), (0.1, 0.2, 0.7), (0.2, 0.1, 0.7)
balance	(0.3, 0.3, 0.4), (0.4, 0.3, 0.3), (0.3, 0.4, 0.3)

4.1.4.1 Dynamic Weights

This strategy changes the weight combinations based on the cost of each examination assignment. If the assignment of an examination incurred some penalty cost, the weight combination is changed in order to find a lower penalty cost than the current examination. Thus, the new examination ordering based on the chosen weight combination is obtained. A new difficult examination is chosen and evaluated whether or not it should be accepted for the time-slot assignment based on penalty cost. The implemented strategy is described in Algorithm 9 below. Let G be the set of heuristic groups. During the iteration, the process is started by choosing the weight combination randomly within HighSD. The weight from the HighSD is chosen because, as explained in Chapter 3, the saturation degree could give better ordering compared with other graph colouring heuristics. It can be observed that the weight combination from the HighSD group still utilised the normalised value of the largest degree heuristic and heuristic modifier but with a smaller contribution.

The penalty cost is calculated for an examination found to have the best time-slot. If the best found time-slot incurred some penalty cost, then the weight combination is changed by choosing randomly from the next successive heuristic group in the list in G. At this stage, only the $difficulty\_score_i(t)$ of examinations within the top-window size is calculated and ordering is performed only for examinations within the top-window size in order to reduce the running time. After calculating the normalised value of examination i and reordering of examinations, the examination with the highest $difficulty\_score$ is taken for further analysis. The process continues until zero penalty cost is found. Once all G has been visited and still no zero penalty cost is found, then the size of the top-window is doubled and the weight changes are repeated once more within the heuristic group in G. If still no zero penalty cost is found, then the examination with the lowest penalty is chosen and is assigned to the best time-slot. The time-slot assignment is implemented as described in Chapter 3 where the least penalty time-slot is chosen for the time-slot assignment. If more than one least penalty time-slot is found then they are chosen randomly.

4.1.4.2 Linear Weights

The strategy is to automatically change the weight value linearly as the number of iterations is increased. This strategy involves only the weight changes of SD and HM. As the process starts, the weight of SD is set to the highest value, while the weight of HM is set to the lowest. On the other hand, the weight value of LD remains constant throughout the iterations. The aim is to change only the weight of SD and HM as both heuristics make significant changes to the examinations ordering. The reason for setting the weight of HM to the lowest is because, at the beginning of the iteration, the HM value is zero and no examinations are considered cannot be scheduled. The HM value contributes no significant changes to examination ordering at that time. As the number of iterations increased, the HM value of some examinations might increase due to the possibility of cannot be scheduled during the iterations. Moreover, our previous study showed that dynamic ordering of saturation degree can contribute to good examination ordering and it is the reason why the weight of SD is set as the highest weight at the start of iterations. As shown in Algorithm 10, at each successive $k * counter$ iterations,

Algorithm 9 Dynamic change of weights during the timetable construction

```
$G = \{HighLD, HighSD, HighHM, Balance\}$
for $t = 1$ to number of iterations do
  Choose weight combination randomly from the group of HighSD
  for $i = 1$ to number of examinations do
    for $j = i$ to number of examinations do
      Calculate the normalise value of chosen heuristics: $choosemax(heurmodN_j(t))$, $HeuristicN_{LD,j}$, $HeuristicN_{SD,j}$ with weight value
      Calculate the $difficulty\_score_j(t)$ for each examination according to the chosen heuristics using equation 4.1
    end for
    Sort($difficulty\_score_i(t)$) in decreasing order
    Calculate penalty-cost for assigning $i$ to the best time-slot
    if penalty-cost of $i > 0$ then
      Change to next group of weight and choose weight combinations randomly
      Calculate the normalise value for examinations within top-window size and order examinations
      if All $G$ has been applied and penalty cost of $i > 0$ then
        Increase the top-window size * 2
        Calculate the normalise value for examinations within top-window size and order examinations
      else
        Choose the lowest penalty-cost and assigned $i$ to lowest penalty time-slot
      end if
    else
      Assign $i$ to the best time-slot
    end if
    if $i$ cannot be scheduled then
      Increase and modify heuristic modifier of $i$ (refer Subsection 3.1.2)
    end if
    Update the Saturation_degree
  end for
  Evaluate solution, store if it is the best found so far
end for
```

the weight values of SD, w_{SD} and HM, w_{HM} are simultaneously changed. The weight value of SD, w_{SD} is decreased by 0.05 while the weight value of HM, w_{HM} is increased by 0.05. Timetables are constructed and evaluated for k time before proceeding to the next weight changes. The process continues until the weight of SD, w_{SD} reaches the minimum value and the weight of HM, w_{HM} reaches the maximum value.

4.1.4.3 Reinforcement Learning

Reinforcement learning is a learning mechanism that interacts with the behaviour of the environment by giving negative and positive rewards based on the performance of an environment and it is achieved by training a learning agent (Sutton and Barto,

Algorithm 10 Linear change of weights during the timetable construction

```
Set the weight combination $w_{LD} = 0.1$, $w_{SD} = 0.85$ and $w_{HM} = 0.05$
Set $counter = 1$
if Linear increment of $w_{HM}$ then
  Set $k = (numberofiterations/m)$ where, $m$ = the number of weight changes that
  occurred during the iteration in order to achieve the highest value of $w_{HM}$
end if
for $t = 1$ to number of iterations do
  //Linear weight changes
  if $t = k * counter$ then
    change the weight value with 0.05 decrement to $w_{SD}$ and 0.05 increment to $w_{HM}$

    Increase counter by one
  end if
  for $i = 1$ to number of examinations do
    for $j = i$ to number of examinations do
      Calculate the normalise value of chosen heuristics: $choosemax(heurmodN_j(t))$,
      $HeuristicN_{LD,j}$, $HeuristicN_{SD,j}$ with weight value
      Calculate the $difficulty\_score_j(t)$ for each examination according to the cho-
      sen heuristics using equation 4.1
    end for
    Sort($difficulty\_score_i(t)$) in a decreasing order
    if $i$ can be scheduled then
      Schedule $i$ in the time-slot with the least penalty
      In the case of the availability of multiple time-slots with the same penalty,
      choose one randomly
    else
      Increase and modify heuristic modifier of $i$ (refer Subsection 3.1.2)
    end if
    Update the Saturation_degree
  end for
  Evaluate solution, store if it is the best found so far
end for
```

1998). The reinforcement learning method involved three main concepts namely the environment, the actions and the reward. The environment allows the learning agent to behave at a defined time, the actions indicates the desirable state of an environment and the reward gives feedback of agent's action in the environment. The approach was successfully implemented in scheduling problems such as nurse rostering (Burke et al., 2003d), parallel machines job shop scheduling (Martinez et al., 2010), wake-up scheduling (Mihaylov et al., 2010), resource constrained project scheduling (Gersmann and Hammer, 2003), examination timetabling (Özcan et al., 2010) and course timetabling (Obit et al., 2009). Burke et al. (2003d)) used a hyper-heuristic method incorporated with tabu search as high level heuristic and a number of low level heuristic that oppose each other based on the rules motivated by reinforcement learning principal, while study by Martinez et al. (2010) introduced value iteration strategy and policy iteration strategy as their reinforcement learning method implemented on the parallel machines job shop scheduling problem. In a wake-up scheduling problem, Mihaylov et al. (2010) presented a decentralized reinforcement learning algorithm. Meanwhile, Gersmann and Hammer (2003) implemented a reinforcement learning approach to resource constrained project scheduling using rout-algorithm combined with support vector machine. Study by Özcan et al. (2010) chose the low-level heuristics based on a reward and punishment scheme known as 'utility value' where this learning mechanism employed an adaptation scheme that was based on the move acceptance by a remembrance mechanism i.e. by remembering or forgetting of the utility value. The forgetting mechanism creates the upper and lower bounds of the utility value. This strategy was combined with the great deluge algorithm as a move acceptance criterion. In a course timetabling problem, Obit et al. (2009) employed a reinforcement learning method with a non-linear great deluge acceptance criteria as a strategy to choose low level heuristics within the hyper-heuristic framework. For further details on the application of reinforcement learning within the hyper-heuristic approach, see Section 2.2.5.

In this study, the strategy uses a reinforcement learning mechanism to punish and reward each weight combination by giving them scores. As illustrated in Algorithm 11, the punishment is based on the performance of each weight combination. If the chosen weight combination can improve the current best solution, then a reward is given by increasing the score by one. On the other hand, if no improvement occurs then a penalty is given to the chosen weight combination by decreasing the score by one. At first, all the score values for each weight combination are initialised as 5 and are decreased or increased at each iteration until they reach the ceiling level. The score value of each weight combination never exceed the value of 10 and the value remains constant even though more improvement occurred in successive iterations. However, the score value of weight combination never less than 0 if no improvement occurs in successive iterations. The next weight combination to be used in the next iteration is chosen based on the best weight combination that has the best score value. In the event that there is more than one best score, then the best score is chosen randomly.

4.2 Experiments

In the experiment described, two benchmark problems were tested. Due to the stochastic nature of the proposed approaches, fifty different timetables were constructed for each dataset from the Toronto and ITC2007. Various combinations of heuristics and heuristic modifiers were considered in order to determine and compare the performance of the proposed approaches. Different weights were also assigned to each heuristic and heuristic

Algorithm 11 Changing the weight combination based on reinforcement learning during the timetable construction

```
$G = \{All\_weight\_combinations\}$
**for** $t = 1$ to number of iterations **do**
   Choose weight combination randomly
   Set all $score_{WeightCombination} = 5$
   **for** $i = 1$ to number of examinations **do**
      **for** $j = i$ to number of examinations **do**
         Calculate the normalise value of chosen heuristics: $choosemax(heurmodN_j(t))$, $HeuristicN_{LD,j}$, $HeuristicN_{SD,j}$ with weight value
         Calculate the $difficulty\_score_j(t)$ for each examination according to the chosen heuristics using equation 4.1
      **end for**
      Sort($difficulty\_score(t)$) in a decreasing order
      **if** $i$ can be scheduled **then**
         Schedule $i$ in the time-slot with the least penalty
         In the case of the availability of multiple time-slots with the same penalty, choose one randomly
      **else**
         Increase and modify heuristic modifier of $i$ (refer Subsection 3.1.2)
      **end if**
      Update the Saturation_degree
   **end for**
   Evaluate solution of current_cost
   **if** current_cost < best_cost **then**
      store the best found so far
      $score_{WeightCombination}$ is increased by 1
      **if** $score_{WeightCombination} > 10$ **then**
         $score_{WeightCombination} = 10$
      **end if**
   **else**
      $score_{WeightCombination}$ is decreased by 1
      **if** $score_{WeightCombination} < 0$ **then**
         $score_{WeightCombination} = 0$
      **end if**
   **end if**
   Choose weight combination that has highest $score_{WeightCombination}$
   If more than one has equal highest score then choose randomly
**end for**
```

modifier with the total weight equal to one for each combination. The stopping condition for this approach was set to 2000 iterations for the Toronto benchmark datasets, while the experiment for ITC2007 was based on the running time given in the competition. The best combination of heuristics and heuristic modifiers is identified based on the best solution obtained. The best penalty value obtained from fifty runs is highlighted in bold for each problem instance. The proposed approach is referred to as the Adaptive Linear Combination (ALC) from this point onwards.

4.2.1 Results of Experiments

4.2.1.1 Toronto

The results of the experiments for different combinations of graph colouring heuristics are provided in Table 4.4. The results show the best penalty value obtained from fifty runs for different combinations of heuristics. The comparison shows that the combination of LDSDHM performed the best with ten out of thirteen datasets and one equal with SDHM, while SDHM obtained best results for two datasets. This circumstance shows that by considering information from more than one parameter simultaneously, the new difficulty measure can be obtained and at the same time a new ordering of examinations can be generated. It can be seen that the single SD performed well in comparison with the single LD and this may be because of the dynamic nature of this heuristic. The single HM also performed well and obtained the best results for six out of thirteen datasets when compared with the other single heuristics.

Table 4.5 illustrates the combination of weights and algorithmic approaches for the best results obtained from the experiments. It shows that most of the best results are obtained using the dynamic heuristic modifier. The value of the dynamic heuristic modifier is updated by finding the highest value of the heuristic modifier each time the assignment process is completed. Taking the shuffling strategy into account, best results are obtained for ten out of thirteen datasets using the top-window strategy, while random preference works more effectively with three out of thirteen datasets. As observed in the table, the weight for HM is the highest for six of the datasets, while four of the datasets obtained the best result with high weight value for SD and the other three datasets performed well with LD as the highest value of weight.

The investigation into different strategies of weight changes is depicted in Table 4.6. As can be seen, the dynamic and linear weight approaches performed almost the same with six and seven best quality solutions respectively on the Toronto benchmark datasets. On the other hand, the reinforcement learning approach obtained no best result for this

Table 4.4: Comparison of single and combination of heuristics (LD = largest degree; SD = saturation degree; HM = heuristic modifier)

Problem	LD	SD	HM	LDSD	LDHM	SDHM	LDSDHM	Best
car91	5.32	5.26	5.43	5.22	5.25	5.18	**5.12**	5.12
t(s)	468	387	888	357	642	389	368	
car92	4.61	4.58	4.63	4.55	4.49	4.42	**4.41**	4.41
t(s)	259	22	487	208	365	222	215	
ears83 I	38.41	39.83	39.52	38.62	38.47	39.06	**36.91**	36.91
t(s)	27	24	25	24	29	24	25	
hec92 I	11.7	11.68	11.72	11.52	11.52	11.42	**11.31**	11.31
t(s)	4	4	3	4	4	5	4	
kfu93	15.24	14.97	15.53	14.97	15.08	**14.75**	14.93	14.75
t(s)	106	234	422	127	128	236	137	
lse91	12.23	11.98	11.79	11.55	11.64	11.51	**11.41**	11.41
t(s)	72	97	260	75	77	98	80	
pur93 I	5.93	6.05	6.42	5.93	5.95	5.92	**5.87**	5.87
t(s)	229	361	1586	290	821	877	920	
rye92	10.25	9.89	9.76	9.95	9.65	**9.61**	9.63	9.61
t(s)	131	202	521	172	136	194	156	
sta83 I	158.63	158.08	157.75	157.84	157.97	157.77	**157.52**	157.52
t(s)	10	14	11	9	10	14	11	
tre92	3.61	3.67	3.67	3.6	3.59	**3.54**	**3.54**	3.54
t(s)	358	292	850	283	416	295	302	
uta92 I	27.14	26.92	26.79	26.79	26.55	26.27	**26.25**	26.25
t(s)	12	20	24	13	13	19	14	
ute92	9.25	8.94	8.85	8.94	8.76	8.81	**8.76**	8.76
t(s)	45	43	55	42	53	43	41	
yor83 I	41.88	40.96	41.48	40.73	41.1	40.08	**39.67**	39.67
t(s)	22	25	26	22	26	24	22	

problem. This is may be due to that the learning mechanism in this approach may require more input or information in learning the behavior of the implemented problem. The success of the linear weight approach may be due to the heuristic modifier that is adapted from time to time. As the number of iterations increased at each successive $k * counter$ iterations, the weight value of the heuristic modifier is increased with 0.05 increment. This is due to the number of violated examinations that may occasionally increase and have a significant effect when the weight of the heuristic modifier is gradually increased. In these circumstances, increasing the weight value of the heuristic modifier can significantly change the examination ordering, the weight value of the saturation degree heuristic also decreases synchronously. On the other hand, the dynamic weight approach also appears to work well. The incorporation of assigning examination cost helps to change the weight combination value. However, this approach increased the

Table 4.5: The combination of weights and algorithmic approaches for the Toronto benchmark datasets (LD = largest degree; SD = saturation degree; HM = heuristic modifier; St = static; Dy = dynamic; REQ = random preference; TW = top-window; AD = additive; EX = exponential; w = weight)

Problem	Best	wLD	wSD	wHM
car92 I	4.41 {LDSDHM, Dy, TW(4), AD}	0.2	0.3	0.5
car91 I	5.12 {LDSDHM, Dy, REQ, EX}	0.1	0.2	0.7
ears83 I	36.91 {LDSDHM, St, TW(4), AD}	0.3	0.1	0.6
hec92 I	11.31 {LDSDHM, Dy, TW(3), AD}	0.2	0.5	0.3
kfu93	14.75 {SDHM, Dy, TW(4), EX}	0.0	0.1	0.9
lse91	11.41 {LDSDHM, Dy, REQ, EX}	0.1	0.5	0.4
pur93 I	5.87 {LDSDHM, St, TW(4), AD}	0.2	0.6	0.2
rye92	9.61 {SDHM, Dy, REQ, EX}	0.0	0.1	0.9
sta83 I	157.52 {LDSDHM, Dy, REQ, EX}	0.5	0.4	0.1
tre92	8.76 {LDSDHM, Dy, REQ, EX}	0.8	0.1	0.1
uta92 I	3.54 {LDSDHM, Dy, REQ, EX}	0.2	0.2	0.6
ute92	26.25 {LDSDHM, Dy, REQ, EX}	0.8	0.1	0.1
yor83	39.67 {LDSDHM, Dy, REQ, EX}	0.1	0.8	0.1

running time as on each occasion no suitable weight combination is found, a new ordering is performed and a new calculation cost of assigning examinations within top-window size to time-slot is obtained.

Tables 4.7, 4.8 and 4.9 report the comparison of results for the thirteen problem instances of the Toronto benchmark for three different groups of approaches i.e. constructive heuristic, hyper-heuristic and other improvement. The comparison with constructive approaches in Table 4.7 shows that the ALC approach obtained no best result. However, some results, such as hec92 I, sta83 I, ute92 and yor83 I, are very close to the best for constructive approaches and were ranked as the second best. Further, based on the overall average rank, the ALC approach is the second best approach after that proposed by Burke et al. (2010e). Comparison with other hyper-heuristic approaches shows that the ALC approach obtained two best results out of thirteen problem instances and is placed as the third best approach. It can be observed that some of the hyper-heuristic approaches shown in Table 4.8 have incorporated a two-phase algorithm, i.e. constructive and improvement. The ALC approach is purely a constructive algorithm that constructs the examination timetables using heuristic ordering. On the other hand, the comparison with other improvement approaches indicates that most of the results from the ALC approach are far from the best results. Nevertheless, some are better than other improvement approaches such as car92 and sta83 I.

In order to identify the difference in solution quality when using various top-window sizes and different groups of weight combination, a two-way analysis of variance is performed.

Table 4.6: Comparison of ALC with different strategies of weight changes for the Toronto benchmark datasets (ALC = adaptive linear combination approach, the bold entries indicate the best results for given strategies)

Problem	Best of ALC	Dynamic weights	Linear weights	Reinforcement learning
car91	5.12	5.33	**5.13**	5.22
t(s)	368	538	273	330
car92	4.41	**4.43**	4.60	4.61
t(s)	215	367	199	176
ear83 I	36.91	**39.02**	39.71	39.76
t(s)	25	41	13	16
hec92 I	11.31	**11.50**	11.91	12.00
t(s)	4	11	2	2
kfu93	14.75	15.70	**15.61**	15.90
t(s)	236	140	78	89
lse91	11.41	12.19	**11.78**	12.34
t(s)	80	96	44	48
pur93 I	5.87	**6.27**	6.29	6.51
t(s)	120	1824	1264	1145
rye93	9.61	**10.40**	10.55	10.51
t(s)	194	140	80	91
sta83 I	157.52	158.50	**158.35**	158.93
t(s)	11	15	9	6
tre92	8.76	9.11	**9.03**	9.11
t(s)	41	82	42	29
uta92 I	3.54	3.63	**3.6**	3.62
t(s)	302	445	210	280
ute92	26.25	**27.48**	28.35	27.91
t(s)	14	13	7	8
yor83 I	39.67	42.83	**42.77**	43.55
t(s)	22	32	11	15

From the statistical analysis, $F_{(31,58156)} = 18.750$ and $\rho(0.000) < 0.05$, it is clear that there are significant differences to solution quality when different top-window sizes and groups of weight combination are employed. Table 4.10 illustrates the effect of different top-window sizes. In most cases different top-window sizes performed significantly differently to one another. However, the top-window size 2 is not significantly different from size 3, while size 3 is not significantly different from sizes 2 and 4.

As stated in Section 4.1.4, the weight combinations are divided into four different groups based on the heuristic contribution. The initial test of the weight combination reveals that there is only little difference when using different types of weight combination. For instance, the weight combination of (0.1, 0.1, 0.8) is not very different from the weight combination of (0.2, 0.1, 0.7) in terms of the performance of solution quality since these

TABLE 4.7: Comparison of ALC with different constructive approaches for the Toronto benchmark datasets (ALC = adaptive linear combination approach, ( ) = rank value, Av. Rank = average rank)

Problem	[1]	[2]	[3]	[4]	[5]	[6]	ALC
car91	7.1 (7)	**4.97 (1)**	5.45 (6)	5.29 (5)	5.03 (2)	5.18 (4)	5.12 (3)
car92	6.2 (7)	4.32 (2)	4.5 (5)	4.54 (6)	**4.22 (1)**	4.44 (4)	4.41 (3)
ears83 I	36.4 (4)	36.16 (3)	36.15 (2)	37.02 (6)	**36.06 (1)**	39.55 (7)	36.91 (5)
hec92 I	**10.8 (1)**	11.61 (4)	11.38 (3)	11.78 (6)	11.71 (5)	12.2 (7)	11.31 (2)
kfu93	**14 (1)**	15.02 (4)	14.74 (2)	15.8 (6)	16.02 (7)	15.46 (5)	14.75 (3)
lse91	**10.5 (1)**	10.96 (3)	10.85 (2)	12.09 (7)	11.15 (4)	11.83 (6)	11.41 (5)
pur93 I	**3.9 (1)**	- (5.5)	- (5.5)	- (5.5)	- (5.5)	4.93 (2)	5.87 (3)
rye92	**7.3 (1)**	- (6.5)	- (6.5)	10.38 (5)	9.42 (2)	10.04 (4)	9.61 (3)
sta83 I	161.5 (6)	161.9 (7)	**157.21 (1)**	160.4 (4)	158.86 (3)	160.5 (5)	157.52 (2)
tre92	9.6 (7)	8.38 (2)	8.79 (6)	8.67 (3)	**8.37 (1)**	8.71 (4)	8.76 (5)
uta92 I	3.5 (4)	**3.36 (1)**	3.55 (6)	3.57 (7)	3.37 (2)	3.49 (3)	3.54 (5)
ute92	**25.8 (1)**	27.41 (4)	26.68 (3)	28.07 (6)	27.99 (5)	29.44 (7)	26.25 (2)
yor83 I	41.7 (5)	40.77 (4)	42.2 (7)	39.8 (3)	**39.53 (1)**	42.19 (6)	39.67 (2)
Av. Rank	3.54 (3)	3.62 (4)	4.23 (5)	5.35 (7)	**3.04 (1)**	4.92 (6)	3.31 (2)

[1]-Carter and Laporte (1996); [2]-Burke and Newall (2004); [3]-Qu and Burke (2007); [4]-Asmuni et al. (2009); [5]-Burke et al. (2010e); [6]-Pais and Burke (2010)

TABLE 4.8: Comparison of ALC with different hyper-heuristics approaches for the Toronto benchmark datasets (ALC = adaptive linear combination approach, ( ) = rank value, Av. Rank = average rank)

Problem	[7]	[8]	[9]	[10]	[11]	[12]	ALC
car91	5.37 (7)	5.36 (6)	**4.97 (1)**	5.16 (3)	5.17 (4)	5.19 (5)	5.12 (2)
car92	4.67 (7)	4.53 (6)	4.28 (2)	**4.16 (1)**	4.32 (4)	4.31 (3)	4.41 (5)
ears83 I	40.18 (7)	37.92 (6)	36.86 (4)	35.86 (3)	**35.7 (1)**	35.79 (2)	36.91 (5)
hec92 I	11.86 (4)	12.25 (7)	11.85 (3)	11.94 (6)	11.93 (5)	**11.19 (1)**	11.31 (2)
kfu93	15.84 (7)	15.2 (5)	14.62 (2)	14.79 (4)	15.3 (6)	**14.51 (1)**	14.75 (3)
lse91	- (7)	11.33 (4)	11.14 (2)	11.15 (3)	11.45 (6)	**10.92 (1)**	11.41 (5)
pur93 I	- (5)	- (5)	**4.73 (1)**	- (5)	- (5)	- (5)	5.87 (2)
rye92	- (5)	- (5)	9.65 (2)	- (5)	- (5)	- (5)	**9.61 (1)**
sta83 I	157.38 (2)	158.19 (4)	158.33 (5)	159 (6)	159.05 (7)	**157.18 (1)**	157.52 (3)
tre92	**8.39 (1)**	8.92 (7)	8.48 (2)	8.6 (4)	8.68 (5)	8.49 (3)	8.76 (6)
uta92 I	- (7)	3.88 (6)	3.4 (2)	3.42 (3)	**3.3 (1)**	3.44 (4)	3.54 (5)
ute92	27.6 (3)	28.01 (5)	28.88 (7)	28.3 (6)	28 (4)	26.7 (2)	**26.25 (1)**
yor83 I	- (7)	41.37 (6)	40.74 (4)	40.24 (3)	40.79 (5)	**39.47 (1)**	39.67 (2)
Av. Rank	5.31 (6)	5.54 (7)	2.85 (2)	4.00 (4)	4.46 (5)	**2.62 (1)**	3.23 (3)

[7]-Kendall and Mohd Hussin (2005a); [8]-Burke et al. (2007); [9]-Pillay and Banzhaf (2009); [10]-Qu and Burke (2009); [11]-Qu et al. (2009a); [12]-Burke et al. (2010f)

TABLE 4.9: Comparison of ALC with different improvement approaches for the Toronto benchmark datasets (ALC = adaptive linear combination approach, ( ) = rank value, Av. Rank = average rank)

Problem	[13]	[14]	[15]	[16]	[17]	[18]	[19]	ALC
car91	5.1 (4)	**4.5** (1)	5.4 (7)	5.2 (6)	6.6 (8)	4.6 (2)	4.8 (3)	5.12 (5)
car92	4.3 (5)	3.93 (2)	4.2 (4)	4.4 (6)	6 (8)	**3.9** (1)	4.1 (3)	4.41 (7)
ears83 I	35.1 (7)	33.71 (3)	34.2 (4)	34.9 (5)	**29.3** (1)	32.8 (2)	34.92 (6)	36.91 (8)
hec92 I	10.6 (5)	10.83 (7)	10.4 (4)	10.3 (3)	**9.2** (1)	10 (2)	10.73 (6)	11.31 (8)
kfu93	13.5 (3)	13.82 (6)	14.3 (7)	13.5 (3)	13.8 (5)	**13** (1)	13 (1)	14.75 (8)
lse91	10.5 (6)	10.35 (5)	11.3 (7)	10.2 (4)	**9.6** (1)	10 (2)	10.01 (3)	11.41 (8)
pur93 I	- (6)	- (6)	- (6)	- (6)	**3.7** (1)	- (6)	4.73 (2)	5.87 (3)
rye92	8.4 (2)	8.53 (3)	8.8 (5)	8.7 (4)	**6.8** (1)	- (8)	9.65 (7)	9.61 (6)
sta83 I	157.3 (3)	158.35 (7)	157 (2)	159.2 (8)	158.2 (5)	**156.9** (1)	158.26 (6)	157.52 (4)
tre92	8.4 (4)	7.92 (3)	8.6 (6)	8.4 (4)	9.4 (8)	7.9 (2)	**7.88** (1)	8.76 (7)
uta92 I	3.5 (5)	**3.14** (1)	3.2 (2)	3.6 (8)	3.5 (5)	3.2 (2)	3.2 (2)	3.54 (7)
ute92	25.1 (3)	25.39 (5)	25.3 (4)	26 (6)	**24.4** (1)	24.8 (2)	26.11 (7)	26.25 (8)
yor83 I	37.4 (7)	36.53 (6)	36.4 (5)	36.2 (2)	36.2 (2)	**34.9** (1)	36.22 (4)	39.67 (8)
Av. Rank	4.62 (5)	4.23 (4)	4.85 (6)	5.00 (7)	3.62 (2)	**2.46 (1)**	3.92 (3)	6.69 (8)

[13]-Merlot et al. (2003); [14]-Yang and Petrovic (2004); [15]-Côté et al. (2005); [16]-Abdullah et al. (2007); [17]-Caramia et al. (2008); [18]-Burke et al. (2010a); [19]-Turabieh and Abdullah (2011).

Table 4.10: The effect of different size of top-window for LDSDHM for the Toronto benchmark datasets ($\neq$ = inequality; $\simeq$ = approximately equal)

Size	Effect	Size
2	$\neq$	(4, 5, 6, 7, 8 and 9 ($p = 0.000$))
2	$\simeq$	(3 ($p = 0.110$))
3	$\neq$	(5, 6, 7, 8 and 9 ($p = 0.000$))
3	$\simeq$	(2 ($p = 0.110$)) and (4 ($p = 0.089$))
4	$\neq$	(2, 5, 6, 7, 8 and 9 ($p = 0.000$))
4	$\simeq$	(3 ($p = 0.089$))
5	$\neq$	(2, 3, 7, 8 and 9 ($p = 0.000$)), (4 ($p = 0.018$)) and (6 ($p = 0.025$))
6	$\neq$	(2, 3, 4, 8 and 9 ($p = 0.000$)), (5 ($p = 0.025$)) and (7 ($p = 0.001$))
7	$\neq$	(2, 3, 4, 5, 8 and 9 ($p = 0.000$)) and (6 ($p = 0.001$))
8	$\neq$	(2, 3, 4, 5, 6 and 7 ($p = 0.000$)) and (9 ($p = 0.001$))
9	$\neq$	(2, 3, 4, 5, 6 and 7 ($p = 0.000$)) and (8 ($p = 0.001$))

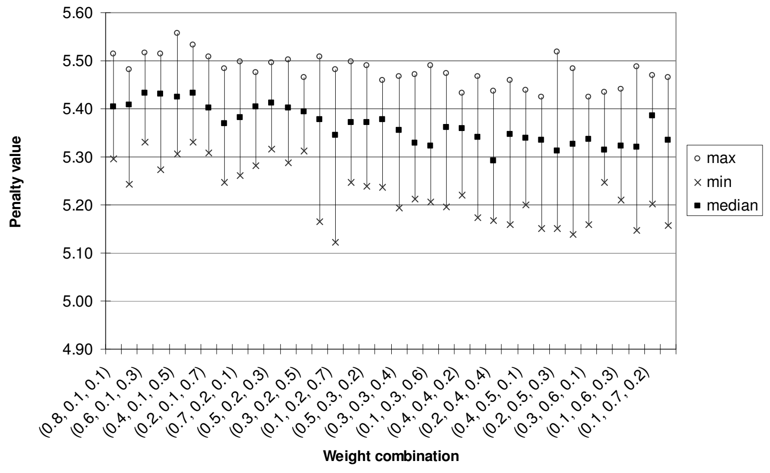
weight combinations are almost identical. In this case, the weight combinations are divided into four different groups, i.e. highLD, highSD, highHM and balance. The details of weight combinations of each group are depicted in Table 4.3. The result from the two-way analysis of variance illustrated in Table 4.11 shows that the solution quality of each group is statistically different. In these circumstances, the solution quality that is tested with the weight combination from any different group of heuristics is statistically different.

Table 4.11: The effect of different group of weight combination for LDSDHM to solution quality for the Toronto benchmark datasets ($\neq$ = inequality)

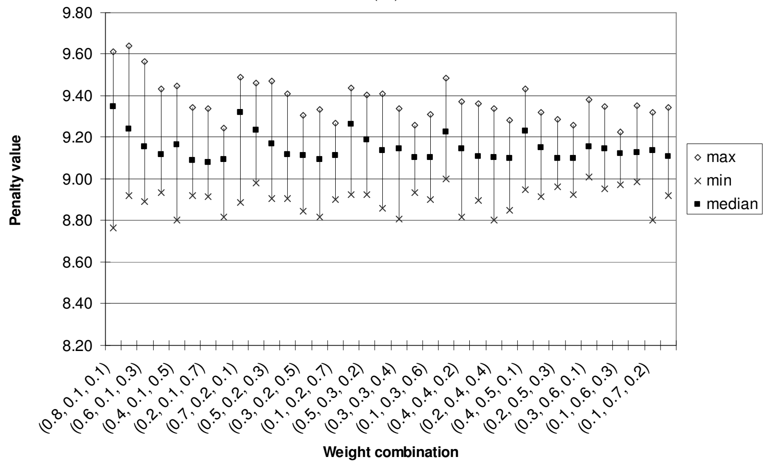
Group	Effect	Group
highLD	$\neq$	(highSD, highHM and balance ($p = 0.000$))
highSD	$\neq$	(highLD, highHM and balance ($p = 0.000$))
highHM	$\neq$	(highLD, highSD and balance ($p = 0.000$))
balance	$\neq$	(highLD, highSD and highHM ($p = 0.000$)

Figure 4.1 illustrates the best performance of LDSDHM for car92 I and tre92 considering all combinations of weight. It demonstrates a pattern in the performance of best solution quality obtained for each combination. By looking at the median value, when the weight value of HM is high enough then the solution quality value rapidly drops. On the other hand, whenever the weight value of HM is gradually decreased, then the solution quality also decreases progressively. Most of the peaks are obtained from the lowest weight value of HM, whilst most of the slumps are obtained from the highest weight value of HM. This shows that the existence of the heuristic modifier in this adaptive linear combination of heuristics has an effect on the solution quality. Furthermore, by using the information from the other two heuristics we have increased the effectiveness of the new ordering.

The results indicate that the combinations are most effective when the weight of HM is very high, while the other heuristics may vary in certain ways.



(a)



(b)

FIGURE 4.1: Best solution quality for each of weight combination of LDSDHM for (a) car91 and (b) tre92 for the Toronto benchmark datasets

Figure 4.2 illustrates the average performance of each top-window size and different groups of weight combination for LDSDHM. It indicates that the group of highLD contributed a higher penalty value at each top-window size while the highSD, highHM and balance are almost identical in terms of penalty value during smaller sizes of top-window performance. However, the average performance for highSD, highHM and balance started to differ when the top-window size is 6 and above. In these circumstances, using a smaller chunk of top-window size with a good choice of weight combinations may lead to a better quality solution.

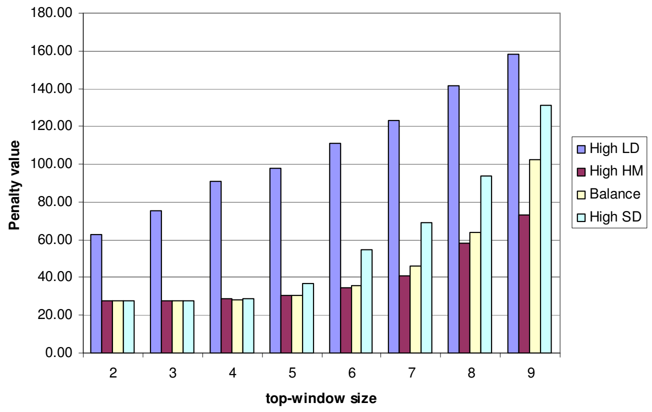


FIGURE 4.2: Average performance of each top-window size and different group of weight combination for LDSDHM for the Toronto benchmark datasets

4.2.1.2 ITC2007

The experiment on the twelve problem instances of ITC2007 was tested with several combinations of weights with only top-window sizes 3 and 5. The heuristic modifier was increased using additive and exponential strategies with dynamic modification of the heuristic modifier value. Table 4.12 illustrates the best penalty value obtained from fifty runs and each solution is provided with information about weight combination and the algorithmic approach. The table reveals diverse patterns of the weight combination of each heuristic for different instances. About half of the problem instances obtained good quality solutions when the weight of the SD is high, while the weights LD and HM are varied in specific ways. Since the ITC2007 benchmark datasets have been tested with only certain parameter settings, unlike the Toronto benchmark datasets, and with time limitations, they might not show the exact pattern of the whole weight behaviour. Moreover, the ITC2007 benchmark datasets represent a capacitated timetabling problem and, therefore, they differ from the Toronto benchmark datasets in terms of various hard and soft constraint requirements.

Table 4.13 illustrates the results of implementing different strategies of weight changes for the ITC2007 benchmark datasets. It is clear that the dynamic weight strategy performed the best among the three strategies with eight problem instances, while the linear weight and reinforcement learning strategies performed the best with three and one problem instances respectively. Although the dynamic weight strategy performed well, this strategy is nevertheless unable to produce a feasible solution for Exam_4. As the ITC2007 instances involve many constraint requirements, the dynamic weight changes might have been an advantage since weight combination is identified based on the cost of each successive assignment. However, as stated previously, this strategy may take up considerable time in calculating the examination assignment cost. It can be

Table 4.12: Different combination of weights and algorithmic approaches for the ITC2007 benchmark datasets (LD = largest degree; SD = saturation degree; HM = heuristic modifier; Dy = dynamic; TW = top-window; AD = additive; EX = exponential; w = weight)

Problem	Best	wLD	wSD	wHM
Exam_1	11060 {LDSDHM, Dy, TW(5), EX}	0.5	0.3	0.2
Exam_2	3133 {LDSDHM, Dy, TW(3), AD}	0.4	0.1	0.5
Exam_3	19098 {LDSDHM, Dy, TW(3), EX}	0.1	0.7	0.2
Exam_4	21309 {LDSDHM, Dy, TW(3), AD}	0.3	0.6	0.1
Exam_5	7975 {LDSDHM, Dy, TW(3), EX}	0.3	0.5	0.2
Exam_6	28330 {LDSDHM, Dy, TW(5), EX}	0.1	0.8	0.1
Exam_7	15912 {LDSDHM, Dy, TW(5), AD}	0.1	0.8	0.1
Exam_8	20066 {LDSDHM, Dy, TW(3), EX}	0.7	0.1	0.2
Exam_9	2165 {LDSDHM, Dy, TW(3), AD}	0.4	0.2	0.4
Exam_10	16516 {LDSDHM, Dy, TW(3), AD}	0.1	0.3	0.6
Exam_11	45873 {LDSDHM, Dy, TW(3), AD}	0.1	0.6	0.3
Exam_12	7465 {LDSDHM, Dy, TW(3), AD}	0.7	0.2	0.1

noted that the experiments on the ITC2007 benchmark datasets followed the running time requirement as stated in the competition rules.

Table 4.13: Comparison of ALC with different strategies of weight changes for the ITC2007 benchmark datasets (ALC = adaptive linear combination approach, *inf.* = infeasible solution, the bold entries indicate the best results for given strategies)

Problem	Best of ALC	Dynamic weights	Linear weights	Reinforcement learning
Exam_1	11060	**11220**	13004	12922
Exam_2	3133	**3176**	3399	3363
Exam_3	19098	**19187**	21245	21147
Exam_4	21309	*inf.*	**20830**	24562
Exam_5	7975	8521	**8201**	8370
Exam_6	28330	**28780**	28960	29060
Exam_7	15912	**16122**	16353	16262
Exam_8	20066	20608	**20480**	20609
Exam_9	2165	2265	2250	**2237**
Exam_10	16516	**16754**	16811	16965
Exam_11	45873	**45124**	50944	66089
Exam_12	7465	**7465**	8348	11488

So far, most of the ITC2007 approaches have concentrated on the multiple phases of solution that construct and improve the solution quality in sequence. The adaptive linear combination approach in this chapter is presented as a constructive approach that iteratively constructs the examination timetable. Table 4.14 shows the comparison of the best penalty values of the adaptive linear combination approach with other approaches from the competition and post-competition. It is clear that the results of the adaptive linear combination approach cannot beat the best results obtained so far and are further from them. However, it is able to produce a feasible solution for all problem instances and is better than some of the approaches from the competition and post-competition. The adaptive linear combination approach is able to produce better results compared with Müller (2009) for Exam_10, Atsuta et al. (2008) for Exam_2, Exam_4 and Exam_6, Pillay (2008) for Exam_1, Exam_4, Exam_6, Exam_7 and Exam_10, Pillay (2010b) for Exam_4 and Exam_6 and also Burke et al. (2010f) for Exam_4, Exam_6 and Exam_8. By contrast, some of the approaches do not produce solutions for some of the problem instances, for example Gogos et al. (2008) for Exam_10 and Exam_12, Atsuta et al. (2008) for Exam_11, De Smet (2008) for Exam_3, Exam_4, Exam_8, Exam_11 and Exam_12 while Pillay (2010b), Burke et al. (2010f) and Turabieh and Abdullah (2011) do not achieve solutions for Exam_9, Exam_10, Exam_11 and Exam_12.

4.3 Conclusion

This chapter proposed an adaptive linear combination of heuristics with a heuristic modifier within the framework of adaptive strategies in order to solve examination timetabling problems. Two graph colouring heuristics with a heuristic modifier were adapted with different weights for each parameter. Each parameter was normalised in order simply to generalise the implemented problem data. A $difficulty\_score$ was used to determine the ordering of examinations and the most difficult examination with the highest $difficulty\_score$ was scheduled first based on two strategies. This approach was tested with single and multiple heuristics with and without a heuristic modifier on the Toronto benchmark datasets while the ITC2007 benchmark datasets were tested with multiple heuristics with heuristic modifier. The results show that by combining multiple heuristics with a heuristic modifier, good solution quality can be obtained. Furthermore, the results from the combination of LDSDHM are comparable with the results of other constructive approaches published in the literature within the Toronto benchmark datasets. The results on the highly constrained ITC2007 benchmark datasets are feasible solutions and some are comparable with previous approaches. In this study, the combination of weight values that are adapted to the heuristics and heuristic modifier could significantly change the examination ordering based on the difficulty_score value. It is found that by changing the weight values of the heuristic and heuristic modifier, good approximate solutions could be obtained. It is also identified that the best top-window size to use for this approach is six and below as the higher value of top-window size could cause

TABLE 4.14: Comparison of ALC with different approaches of the ITC2007 benchmark datasets (ALC = adaptive linear combination approach)

Problem	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	ALC
Exam_1	4370	5905	8006	6670	12035	4370	4633	8559	6235	4775	**4368**	11060
Exam_2	400	1008	3470	623	3074	385	405	830	2974	**385**	390	3133
Exam_3	10049	13862	18622	-	15917	9378	9064	11576	15832	**8996**	9830	19098
Exam_4	18141	18674	22559	-	23582	**15368**	15663	21901	35106	16204	17251	20830
Exam_5	2988	4139	4714	3847	6860	2988	3042	3969	4873	**2929**	3022	7975
Exam_6	26950	27640	29155	27815	32250	26365	25880	28340	31756	**25740**	25995	28330
Exam_7	4213	6683	10473	5420	17666	4138	**4037**	8167	11562	4087	4067	15912
Exam_8	7861	10521	14317	-	16184	7516	**7461**	12658	20994	7777	7519	20066
Exam_9	1047	1159	1737	1288	2055	1014	1071	-	-	**964**	-	2165
Exam_10	16682	-	15085	14778	17724	14555	14374	-	-	**13203**	-	16516
Exam_11	34129	43888	-	-	40535	31425	29180	-	-	**28704**	-	45124
Exam_12	5535	-	5264	-	6310	5357	5693	-	-	**5197**	-	7465

[1]-Müller (2008); [2]-Gogos et al. (2008); [3]-Atsuta et al. (2008); [4]-De Smet (2008); [5]-Pillay (2008); [6]-Müller (2009); [7]-McCollum et al. (2009); [8]-Pillay (2010a); [9]-Burke et al. (2010f); [10]-Gogos et al. (2010a); [11]-Turabieh and Abdullah (2011)

a significant change in the examination ordering. It is, therefore, concluded that this approach is simple and effective, and hence has potential for practical use.

Chapter 5

A Constructive Approach for Examination Timetabling based on Adaptive Decomposition and Ordering

This chapter draws upon the research from previous chapters by further considering the unscheduled examinations obtained during timetable constructions. The work is based on a similar adaptive approach by Qu and Burke (2007) that has made use of a decomposition strategy. A methodology which divides the problem into two sub-problems related to the feasibility of the current assignment is proposed. The same naming convention is adopted from Qu and Burke (2007) for these sets as ‘difficult’ and ‘easy’. In this chapter, the problem is decomposed into difficult and easy sets at each iteration. A timetable is constructed based on the associated heuristic ordering for each set. An additional set of examinations which is located between the difficult and easy sets is also introduced. This is referred to as the boundary set. Several mechanisms associated with the boundary set are described in order to vary the search space of solutions. Section 5.1 presents the details of the proposed approach based on adaptive decomposition and ordering (ADO) for examination timetabling. Section 5.2 discusses the analysis and experimental results. Finally, the conclusion is provided in Section 5.3.

5.1 Automated Decomposition and Ordering of Examinations

The study of constructive approaches within a decomposition strategy is a relatively unexplored area of timetabling. In decomposition approaches, a problem is divided into a smaller sub-problem and the solution is obtained from each sub-problem taking into account the related constraints. Carter and Laporte (1996) stated that this strategy could reduce the processing time. However, it could adversely affect the quality of the final solution because of the drastic reduction of the search space during the scheduling process. In these circumstances, the sub-problems need to be carefully reunited to sustain the optimality of the solution.

One of the decomposition approaches related to mathematical programming method is the Bender decomposition, which separates integer variables and real variables by allowing large problem to be decomposed into block structures in order to achieve for efficiency (Benders, 1962). A number of studies in scheduling problems used this approach for decomposing the problem into smaller problems. Such application can be found in the employee timetabling problem (Detienne et al., 2009), workforce scheduling problem (Benoist et al., 2002) and crew scheduling problem (Mercier et al., 2005). Other approaches related to decomposition within timetabling problems are by Meisels et al. (1994), De Causmaecker et al. (2009) and Burke et al. (2010d). The study by Meisels et al. (1994) focused on solving the binary form of the school timetabling problem by representing it as constraint networks. The problem was solved using graph decomposition. Meanwhile, De Causmaecker et al. (2009) worked on the course timetabling problem by using decomposition approach and focused on the overlapping time-slots and irregular weekly timetable. The approach introduced a 'pillar' structure in order to reduce the number of courses and considered only one constraint at a time before proceeding with the next constraints during the timetabling. The next study, Burke et al. (2010d) proposed a hybrid metaheuristic approach for solving course timetabling problems where it was based on a mix formulation of different sub-problems formulated in integer programming.

The decomposition approach in timetabling is related to splitting a problem into a several smaller sub-problems with the purpose of easing the scheduling process and at the same time aiming for high quality solutions. There are several implementation strategies for splitting the problem within timetabling approaches. Figure 5.1 shows decomposition strategies in timetabling (Figures 5.1 (a) to (c) are adopted from Özcan and Alkan (2007)). Each strategy has a number of stages to be processed to each subset and each subset contains a set of events to be scheduled.

The strategy depicted in Figure 5.1 (a) was implemented by Burke et al. (1996b) in their attempt to solve the examination timetabling problem. During the scheduling process, at each stage, a selected subset was scheduled separately. As the scheduling was completed for the current subset, then the scheduling was continued to the next subset of events. Burke and Newall (1999) implemented the second strategy, as shown in Figure 5.1 (b), which was a union of a scheduled and an unscheduled subset. The scheduled subset was fixed until the end of a phase and the problematic examinations that cause infeasible timetable was solved using a look-ahead approach. The third strategy introduced by Özcan and Alkan (2007) was applied to the course timetabling problem. This was more flexible in that the scheduled subset could be reassigned to a timetable. As depicted in Figure 5.1 (c), the unscheduled subset is incrementally added to the scheduled subset for the assignment process. The three strategies have a fixed size of problem subsets. However, they differ from the fourth approach illustrated in Figure 5.1 (d) where the strategy used a flexible size of subsets depending on the unscheduled events occurring during the timetabling. This strategy, introduced by Qu and Burke (2007) in relation to the examination timetabling problem, considered two subsets to be scheduled, and introduced boundary examinations between the two subsets. The study provides the basis for the proposed approach in this chapter.

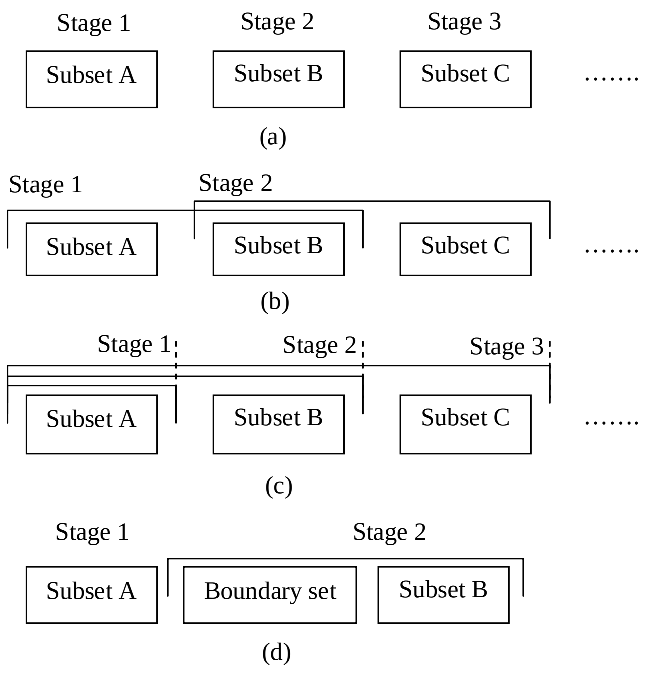


Figure 5.1: Strategies of decomposition in timetabling

Most timetabling approaches in the literature do not make use of information obtained from the process of building an infeasible timetable. The examinations causing the infeasibility of a solution indicates that they are difficult to place and should perhaps be treated in different ways. A general constructive framework as presented in Pseudocode

12 is proposed for solving the examination timetabling problem, based on the adaptive decomposition and ordering of a set of examinations into two sets: difficult and easy.

Let E be the set for all unscheduled examinations. At first, all the examinations from E are considered in the $EasySet$, while there is no examination in the $DifficultSet$ as all examinations are assumed to be easy to schedule at the beginning of the timetabling process. As the timetabling process begins, some of the examinations causing infeasibility are moved to the $DifficultSet$. At each iteration, each $DifficultSet$ and $EasySet$ is ordered based on the chosen heuristic within the sets. The $BoundarySet$ is created within the $EasySet$ where a fixed number of examinations, δ, with higher difficulty value are chosen to be in the $BoundarySet$. This $BoundarySet$ is merged or swapped with the $DifficultSet$ using the chosen strategy. Each examination is scheduled to the least penalty time-slot and if there is more than one least penalty time-slot then they are chosen randomly. If examination e cannot be scheduled it is left unscheduled and is moved to the $DifficultSet$. In case of no improvement to the solution quality for a certain scheduling trial, the shuffling-strategy is employed. The shuffling-strategy aims to shuffle the current best examination ordering so that a new ordering could be obtained.

Algorithm 12 Construction of a timetable based on automated decomposition and ordering of examinations

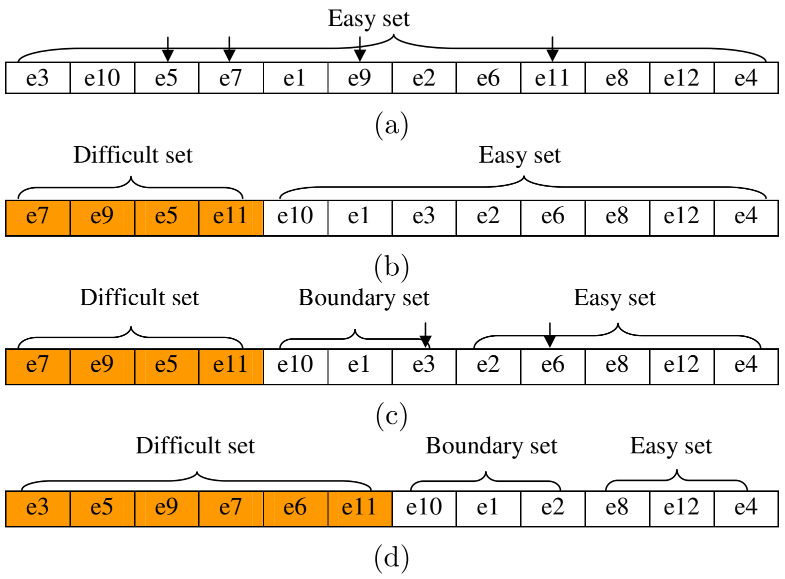
```
$E = \{e1, e2, ., eN\}$
$BoundarySetSize = \delta$
$EasySet = E$; $DifficultSet = \phi$; $BoundarySet = \phi$; $TempSet = \phi$
Initial ordering
for $i = 1$ to number of iterations do
  $OrderExamsWithinSubsets(DifficultSet, EasySet)$
  $BoundarySet = CreateBoundarySet(DifficultSet, EasySet)$
  while there are examinations to be scheduled do
    Consider changing the ordering of examinations using Shuffling-Strategy
    Employ Selection-Strategy to choose an unscheduled exam, $e$
    if $e$ can be scheduled then
      $TempSet = TempSet \cap \{e\}$
      Schedule e in the time-slot with the least penalty
      In the case of the availability of multiple time-slots with the same penalty,
      choose one randomly
    else
      Move exam $e$ to $DifficultSet$
    end if
    $EasySet = TempSet$
  end while
  Evaluate solution, store if it is the best found so far
end for
```

During each iteration, a new solution is constructed from an ordered list of examinations. The difficult set consists of the examinations that cannot be placed into a time-slot within the timetable due to conflicts with other examinations from the previous iteration. These examinations need to be associated with a large penalty imposed on the unplaced examinations. On the other hand, the examinations in the easy set are examinations that cause no violations during the timetabling. Using this approach, all the examinations that contribute to the infeasibility in a solution are given priority. They are moved forward in the ordered list of examinations and are processed first. Such examinations are detected and included in the difficult set at each iteration and a predefined ordering strategy is employed before their successive assignment to the available time-slots. The remaining examinations (that generate no feasibility issues) are placed into the easy set and the original ordering of those examinations is maintained. In order to incorporate a stochastic component for the selection of examinations from the generated ordering, some shuffling strategies are utilised. The following subsections discuss these strategies.

5.1.1 Interaction between Difficult and Easy Sets through a Boundary Set

The ADO approach is investigated using two graph colouring heuristics for generating the initial ordering of examinations. The approach is tested with the largest degree heuristic that orders examinations in a decreasing number of conflicts with each examination and the saturation degree heuristic that dynamically orders the unscheduled examinations based on the number of available time-slots for each examination during the timetable construction. The reason for testing these two graph colouring heuristics is to compare their performance in terms of solution quality and the contribution of infeasible examinations to the size of the difficult set, as they represent static and dynamic ordering of heuristics. Initially, all the examinations are considered to be members of the easy set (as illustrated in Figure 5.2 (a)).

During each iteration, examinations causing infeasibility are identified. Figure 5.2 (a) shows that all such examinations are marked as members of the difficult set to be moved forward towards the top of the list of examinations (Figure 5.2 (b)), while the examinations that caused no violation during the assignment to a time-slot remain in the easy set. In Figure 5.2 (c), the boundary set with a prefixed size within the easy set is created and it is located in between the difficult and the easy sets. Examinations in the boundary set are chosen based on the difficulty value where the most difficult ones in the easy set are chosen to be included in the boundary set. The examinations in the boundary set are merged with the examinations in the difficult set by combining both sets before a reordering is performed. Once the ordering for all examinations within



Easy set
e3 e10 e5 e7 e1 e9 e2 e6 e11 e8 e12 e4
(a)
Difficult set
Easy set
e7 e9 e5 e11 e10 e1 e3 e2 e6 e8 e12 e4
(b)
Difficult set
Boundary set
Easy set
e7 e9 e5 e11 e10 e1 e3 e2 e6 e8 e12 e4
(c)
Difficult set
Boundary set
Easy set
e3 e5 e9 e7 e6 e11 e10 e1 e2 e8 e12 e4
(d)

FIGURE 5.2: (a) All examinations are in easy set in the first iteration and examinations that cause infeasibility are marked; (b) difficult and easy sets after an iteration resulting with an infeasible solution; (c) boundary set with a prefixed size is added to the difficult set after an iteration and reordering is performed; (d) the step in (a) is repeated and the infeasible examinations are placed in the difficult set, the size of difficult set increased

the difficult and boundary sets are performed, each examination is assigned one by one to a feasible time-slot and followed by the assignment of the easy set. In the next iteration, more infeasible examinations are detected and included in the difficult set. Consequently, the size of the difficult set is increased from one iteration to another if there are infeasible examinations.

5.1.2 Swapping the Examinations Between Difficult and Boundary Sets

This strategy shuffles the difficult set and the boundary set by swapping examinations randomly between them. Some examinations causing infeasibility are not necessarily the one that are very difficult to schedule. The infeasibility may happen due to the previous assignment and ordering. This strategy introduces an opportunity for some of the examinations in the difficult set to be chosen later in the timetable. There is also a possibility that the examinations in the boundary set are swapped back to the original set because this process is done randomly. Figure 5.3 illustrates how the swapping of examinations between two sets might take place.

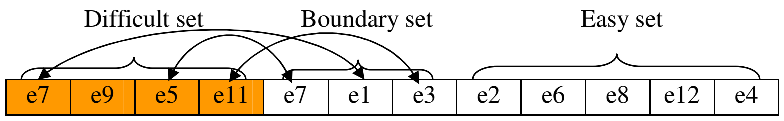


FIGURE 5.3: The boundary set is swapped with the difficult set and is reordered before assigning examinations to the time-slots

5.1.3 Roulette Wheel Selection for Examinations

A roulette wheel selection strategy that incorporates a stochastic element in choosing examinations is utilised before assigning them to the time-slots. If there is no improvement evident within a certain time, a list of examinations of size n is chosen from the ordered list in the difficult set from which an examination is chosen based on a probability. The probabilities of an examination being chosen are calculated based on a score, s_i of each examination in the list of size n. The new size of the difficult set will be the set which includes the size of the boundary set whenever there is improvement to the solution quality. The score value, s_i is a dynamic measure that is obtained from the largest degree and saturation degree values (as in equation (5.1)), where Num_clash_i is the number of examinations in conflict with the examination i, Max_clash is the maximum number of conflicts with all examinations, Sat_degree_i is the saturation degree value for the examination i and Num_slots is the number of time-slots given to the specified problem. The score, s_i for the i^{th} examination measures the difficulty of scheduling it, which combines the saturation degree, Sat_degree_i of the given examination and the number clashing examinations, Num_clash_i. The larger the score, the more difficult it is to schedule the examination. The Num_clash_i value is aligned with this formulation, while Sat_degree_i requires an adjustment, as the saturation degree of an examination gets lower and lower, scheduling it becomes more difficult. In this study, the complement of Sat_degree_i as (max-$number$-of-$time$-$slots$ - $Sat\_degree_i + 1$) is used for the i^{th} examination while computing s_i. Consequently, its initial value is set to 1. This strategy is adopted from Abdul Rahman et al. (2009).

$$s_i = \frac{Num\_clash_i}{Max\_clash} + \frac{Sat\_degree_i}{Num\_slots} \tag{5.1}$$

The probability, p_i of an examination being chosen from n list of examinations is,

$$p_i = \frac{s_i}{\sum_{i=0}^{n-1} s_i} \tag{5.2}$$

A random number from $(0, 1)$ is obtained in order to choose an examination from a list of examinations of size n. An examination with higher score value, s_i will have a greater chance to be chosen.

5.1.4 Comparison of Our Approach to a Previous Study

Qu and Burke (2007) previously proposed an adaptive decomposition approach to construct examination timetables. Their approach started with an initial ordering of examinations using a graph colouring heuristic, namely, saturation degree. A perturbation was made by randomly swapping two examinations in order to obtain a better ordering. Examinations were then decomposed into two sets: difficult and easy.

The initial size of the difficult and easy sets is prefixed as half of the number of examinations in a given problem, as shown in Figure 5.4 (a). At each iteration, the size of the difficult set is modified according to the feasibility of the solution. If the solution is infeasible after the adjustment of the ordering of examinations then the first examination that causes infeasibility (for example, e11) is moved forward for a fixed number of places (for example, five as illustrated in Figure 5.4 (b)). The size of the difficult set is then re-set to the point where the difficult examination is placed. Otherwise, if a feasible solution or an improved solution is obtained, then the size of the difficult set is increased (Figure 5.4 (c)). The approach also introduced boundary examinations where half of the examinations in the difficult set are considered to be in the easy set due to assumption that some examinations within the difficult and easy sets are overlapping.

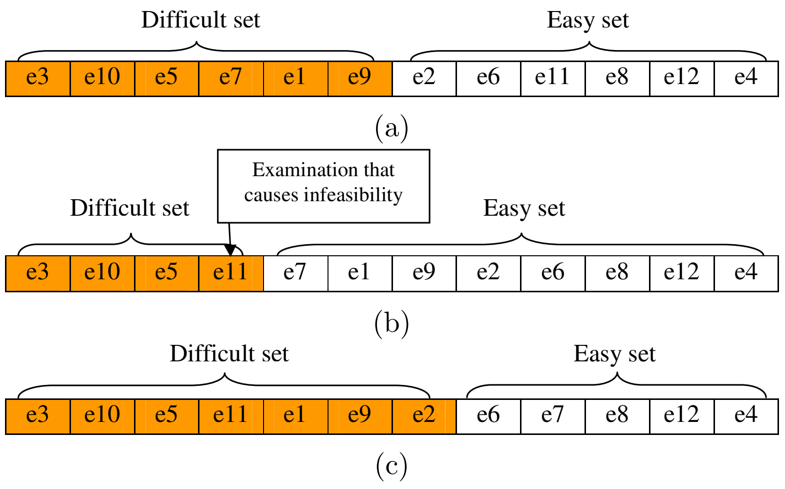


FIGURE 5.4: Difficult and easy sets (a) in the first iteration; (b) after an iteration is over (a) resulting with an infeasible solution; (c) after an iteration is over (a) resulting with a feasible solution

The ADO approach initialises the easy set including all the examinations and the difficult set is formed during each construction phase at each iteration. The size of the difficult set depends on the number of unscheduled examinations that cannot be assigned to any time-slot from all previous iterations. The size of the difficult set never decreases and after a certain number of iterations, the number of examinations in the difficult set might settle. On the other hand, in the previous approach, the size of the difficult set is prefixed and increased when the feasible solution or improved solution is obtained statically. The set is also allowed to shrink. Additionally, the previously proposed approach uses an initial ordering and reorders all the examinations without using a heuristic, which is not the case in the ADO approach. Although the ADO approach used the same approach for reordering examinations in the difficult and easy sets separately, examinations in different sets can be reordered based on a different heuristic at each iteration. The ADO approach also considered the boundary set which contains examinations from the easy set that has higher difficulty value and these examination are considered difficult due to the higher difficulty value within the easy set and they are moved into the boundary set. The examinations in the boundary set can be merged or swapped with the examinations within the difficult set.

5.2 Experiments

Numerical experiments were carried out using Pentium IV 1.86 GHz. Windows machines with 1.97 Gb. memory. The experiments on the Toronto benchmark datasets were performed with twenty-five runs and the stopping condition was set at 2000 iterations in order to be comparable with the experiments conducted by Qu and Burke (2007). The previous study (Abdul Rahman et al., 2010) demonstrated that the increase of the number of iterations produced no significant improvement to the solution quality. Therefore, it was decided to increase the number of runs while reducing the number of iterations. Meanwhile, the experiments for the ITC2007 benchmark datasets were performed with fifty runs in order to be equal to some of the previous implementations. For further detail on the descriptions and the constraints of the Toronto and ITC2007 benchmark datasets see Section 2.5. Two types of heuristic ordering for initialisation were investigated: largest degree (LD) and saturation degree (SD). The difficult set was created using these two initial orders, then reordered with either largest degree or saturation degree. In this study, the same heuristic for initialisation was used to order the examinations in the easy set. The heuristics used in a given approach were denoted by a pair as [*heuristic used for ordering examinations in the difficult set - heuristic used for ordering examinations in the easy set*] from this point onwards.

5.2.1 Parameter Tuning

In order to identify the best parameter setting, the approach was tested on the Toronto benchmark datasets with six different sizes of the boundary set {0, 3, 5, 10, 15, 20}. Figure 5.5 illustrates the experiments on the average cost of overall performance for different sizes of boundary sets with different heuristic orderings for the difficult and easy sets combined with both adding and swapping strategies. Considering overall performance, it appears that, on average, boundary set size 3 is the best for implementation with lower standard deviation compared with other sizes. Based on the T test, it is also statistically significant that boundary set size 3 is different compared with sizes {0, 10, 15, 20} where the *p value* is < 0.05. However, when compared with boundary set size 5, the performance is about the same but boundary set size 3 is still better in terms of the average cost and the standard deviation. In this case, the boundary set size 3 is chosen to be experimented in the ADO approach. Table 5.1 presents the solution quality experiment with boundary set size 3 and tested with thirteen problems of the Toronto benchmark datasets.

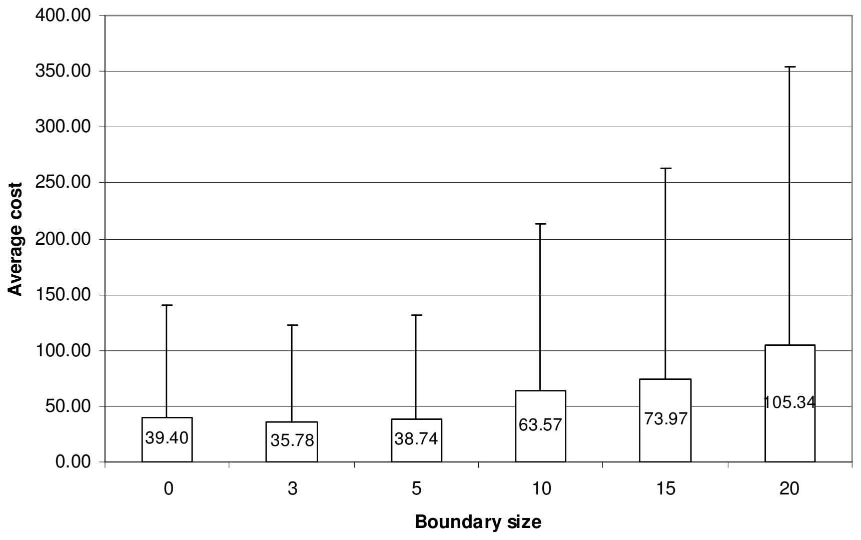


FIGURE 5.5: Average cost of overall performance for all problem instances for different sizes of boundary set of the Toronto benchmark datasets

The utilisation of the shuffling strategy of the Toronto benchmark datasets with roulette wheel selection is also investigated, where different sizes of n examinations are stochastically selected from size $n = \{0, 3, 5, 10, 15\}$. Figure 5.6 shows the different performances of the approach with different sizes of roulette wheel selection. The statistical test revealed that there is a significant difference when incorporating the roulette wheel selection in the approach; the *p value* < 0.05 when comparing size 3 with size {0, 10, 15}. However, size 3 is statistically no different from size 5. In choosing the best setting for the roulette wheel size, it is observed that size $n = 3$ performed best in terms of the

TABLE 5.1: Comparing best solution quality of the Toronto benchmark datasets for (a) [LD-LD], (b) [LD-SD], (c) [SD-LD] and (d) [SD-SD] by adding boundary set into difficult set and swapping examinations between boundary and difficult sets with $\delta = 3$ (av. = average solution quality; std. = standard deviation; t(s) = average running time in seconds) (Bold font indicates the best for different ordering and strategy and bold and italic is the best of all for each problem instance)

		car91	car92	ear82 I	hec92 I	kfu93	pur93 I	lse91	rye93	sta83 I	tre92	ute92	uta92 I	yor83 I
Add the boundary set ($\delta = 3$) into difficult set	(a)	5.69	4.85	41.15	12.66	16.35	6.44	13.44	10.52	160.73	9.46	29.15	3.88	44.70
	av.	5.88	5.10	42.71	13.85	16.87	6.54	14.07	11.16	165.31	9.94	30.12	4.00	725.71
	std.	0.06	0.09	0.68	0.67	0.31	0.04	0.24	0.28	1.42	0.20	0.59	0.05	318.47
	t(s)	22.44	14.16	2.44	0.80	4.40	98.84	3.12	6.72	0.68	3.60	0.84	17.44	2.40
	(b)	5.70	***4.74***	42.27	**12.35**	16.45	6.56	12.87	**10.30**	***159.03***	***9.07***	29.27	3.79	**44.23**
	av.	5.85	5.01	43.88	13.09	17.02	6.66	13.41	10.81	160.42	9.74	30.11	3.91	46.20
	std.	0.07	0.09	0.88	0.39	0.31	0.05	0.19	0.20	0.92	0.27	0.51	0.06	0.71
	t(s)	53.52	31.36	5.04	1.16	8.20	367.52	5.80	12.32	1.28	7.36	1.40	40.36	4.96
	(c)	***5.27***	4.79	***41.12***	12.69	16.11	6.15	**12.44**	10.38	160.98	9.49	28.81	3.73	45.27
	av.	5.60	5.00	43.64	13.39	16.87	6.41	13.43	10.71	163.68	9.85	29.95	3.91	46.88
	std.	0.15	0.08	1	0.41	0.32	0.11	0.33	0.17	1.60	0.16	0.62	0.10	0.97
	t(s)	102.48	59.36	6	1.24	30.96	1437.04	17.36	35.12	2.00	12.24	3.92	80.84	5.00
	(d)	5.41	4.75	42.02	12.70	***16.01***	***6.05***	12.74	10.39	160.05	9.35	**28.63**	***3.72***	44.48
	av.	5.59	4.99	43.73	13.24	16.92	6.36	13.30	10.79	162.23	9.82	29.96	3.91	46.48
	std.	0.12	0.09	0.72	0.37	0.45	0.16	0.29	0.14	1.05	0.19	0.49	0.09	0.62
	t(s)	102.76	59.44	6.20	1.16	31.36	1425.60	17.84	34.88	2.24	12.12	3.92	79.72	5.04
Swap in the boundary ($\delta = 3$) and difficult set	(a)	5.77	4.99	41.83	12.98	16.30	6.44	13.43	10.63	160.55	9.36	28.96	3.89	*inf.*
	av.	5.90	65.16	63.55	113.86	16.97	6.55	34.06	31.27	163.16	10.03	30.19	4	944.7
	std.	0.07	165.79	99.60	203.64	0.29	0.06	99.66	99.98	1.64	0.27	0.65	0.06	249.34
	t(s)	13.72	8.52	1.64	0.48	2.68	61.52	1.92	4.04	0.48	2.20	0.48	10.56	1.48
	(b)	5.77	4.81	42.24	***12.05***	**16.25**	6.45	12.85	***10.24***	**159.62**	9.51	28.88	3.82	44.93
	av.	5.88	5.07	44.10	13.40	17.05	6.63	13.41	10.8	160.86	9.81	29.90	3.96	46.57
	std.	0.07	0.09	0.76	0.54	0.39	0.11	0.31	0.24	0.96	0.16	0.55	0.06	0.80
	t(s)	32.04	18.80	3.08	0.76	4.92	224.76	3.48	7.36	0.76	4.44	0.80	24.08	2.96
	(c)	5.33	**4.75**	42.18	12.53	16.43	**6.09**	***12.41***	10.48	160.29	**9.27**	29.11	**3.77**	***44.19***
	av.	5.63	4.99	43.79	13.15	16.92	6.41	13.30	10.76	162.79	9.87	29.94	3.91	46.55
	std.	0.14	0.09	0.80	0.28	0.26	0.17	0.31	0.14	1.29	0.17	0.37	0.10	1.00
	t(s)	60.68	35.28	3.68	0.76	18.04	865.04	10.36	21.16	2.00	7.40	2.28	47.40	3.04
	(d)	**5.32**	4.81	**41.34**	12.51	16.33	6.14	12.87	10.36	160.29	9.41	***27.75***	3.78	44.94
	av.	5.60	4.98	43.74	13.17	16.88	6.42	13.46	10.72	162.79	9.86	29.84	3.92	46.61
	std.	0.12	0.08	1.05	0.33	0.33	0.17	0.28	0.15	1.29	0.19	0.66	0.10	0.56
	t(s)	60.56	35.40	3.72	0.76	18.12	865.32	10.48	21.32	2.08	7.32	2.32	47.36	3.12

average solution quality and also the standard deviation. The size $n = 3$ is chosen to be experimented with boundary size 3. This analysis has shown that the incorporation of the shuffling strategy improved the performance in terms of the average and the variance of the solution cost. The solution quality for each problem instance of the Toronto benchmark with different setting of heuristic combination of the boundary size $\delta = 3$ and roulette wheel selection size $n = 3$ is presented in Table 5.2.

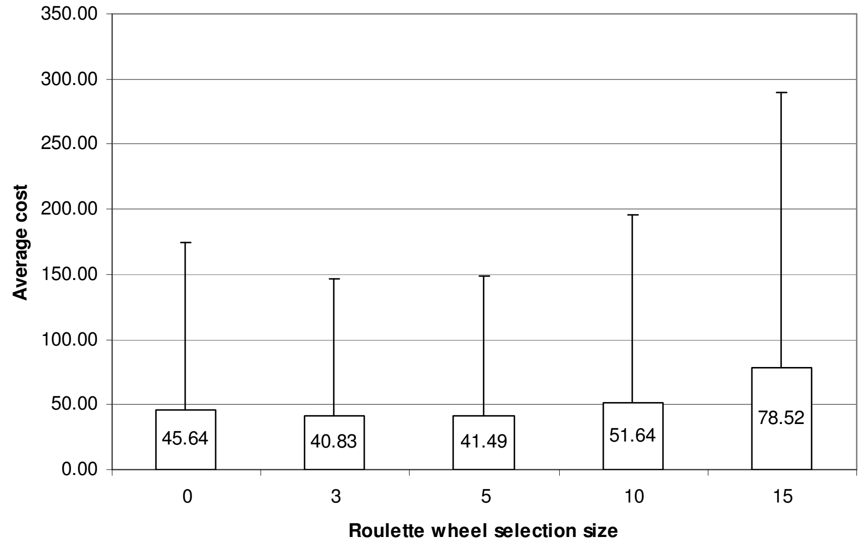


FIGURE 5.6: Average cost of overall performance for all problem instances for different size of roulette wheel selection size of the Toronto benchmark datasets

5.2.2 Best Performance Comparison of Different Strategies

Table 5.1 summarises the experimental results obtained by applying the proposed approach to the Toronto benchmark problem instances with boundary size 3. By looking at the best strategy of this approach, it is observed that the adding boundary set strategy performed better with eight best problem instances compared with the swapping boundary set strategy. Table 5.1 illustrates that the saturation degree based initial solution performed significantly better than the largest degree based initial ordering in terms of average best solutions obtained.

By looking at the best heuristic ordering for the difficult and the easy sets, it is observed that with the boundary set size 3, the adding boundary set strategy performed slightly better with the saturation degree based initial ordering where seven out of the thirteen problem instances performed significantly better than the largest degree based initial ordering. On the other hand, the swapping strategy performed better with the saturation degree based initial ordering with nine out of thirteen problem instances being better when compared with the largest degree initial ordering.

TABLE 5.2: Comparing solution quality of the Toronto benchmark datasets for (a) [LD-LD], (b) [LD-SD], (c) [SD-LD] and (d) [SD-SD] with shuffling strategies of adding the boundary set into the difficult set and swapping examinations between the boundary and difficult sets with $\delta = 3$ and includes roulette wheel selection for examinations with $n = 3$ (av. = average solution quality; std. = standard deviation; t(s) = average running time in seconds) (Bold font indicates the best for different ordering and strategy and bold and italic is the best of all for each problem instance)

		car91	car92	ear82 I	hec92 I	kfu93	pur93 I	lse91	rye93	sta83 I	tre92	ute92	uta92 I	yor83 I
Add the boundary set ($\delta = 3$) into difficult set	(a)	5.75	4.86	41.15	***12.26***	16.27	6.42	12.93	10.72	160.51	9.33	***27.71***	3.91	46.30
	av.	5.80	5.02	42.58	12.77	16.72	6.53	13.41	11.08	161.62	9.91	28.52	3.98	625.53
	std.	0.10	0.10	0.67	0.32	0.27	0.05	0.23	0.29	0.66	0.24	0.72	0.04	276.37
	t(s)	31.92	18.44	3.00	0.72	4.83	225.56	3.48	7.40	0.68	4.36	0.80	23.88	2.96
	(b)	5.74	4.82	41.85	12.44	16.35	6.48	12.77	**10.22**	***158.12***	9.40	28.37	3.82	45.00
	av.	5.82	4.90	42.88	12.99	16.83	6.63	13.14	10.72	159.81	9.68	28.93	3.90	46.08
	std.	0.05	0.09	0.84	0.30	0.25	0.07	0.21	0.17	0.69	0.18	0.49	0.05	0.51
	t(s)	32.56	19.16	3.16	0.76	5	228.08	3.56	7.48	0.80	4.60	0.84	24.36	3.00
	(c)	**5.30**	4.88	42.14	12.43	**16.27**	**6.07**	***12.58***	10.39	161.59	9.37	27.87	3.77	44.44
	av.	5.43	4.95	42.89	12.91	16.58	6.37	13.02	10.59	163.09	9.65	28.31	3.90	45.71
	std.	0.11	0.06	0.74	0.36	0.26	0.19	0.24	0.15	1.20	0.17	0.41	0.10	0.98
	t(s)	60.4	35.08	3.64	0.76	17.00	866.6	10.04	20.84	1.20	7.16	1.96	47.24	3.12
	(d)	5.31	***4.74***	***40.91***	12.36	16.31	**6.07**	12.84	10.40	159.20	***9.30***	27.80	**3.74**	***43.98***
	av.	5.45	4.82	41.93	12.78	16.84	6.36	13.06	10.62	159.84	9.54	28.25	3.89	44.53
	std.	0.10	0.09	0.78	0.31	0.35	0.19	0.30	0.17	1.02	0.21	0.63	0.08	0.86
	t(s)	60.52	35.16	3.60	0.80	17.44	868.84	9.84	20.40	1.20	7.28	2.00	47.24	3.12
Swap in the boundary ($\delta = 3$) and difficult set	(a)	5.74	5.02	42.20	**12.47**	16.23	6.41	12.69	10.61	159.62	9.60	28.54	3.92	inf.
	av.	5.84	65.09	43.89	13.1	16.74	6.54	13.54	11.32	161.01	9.98	29.19	3.99	725.24
	std.	0.10	165.78	1.21	0.40	0.38	0.10	0.48	0.73	0.87	0.16	0.46	0.05	244.22
	t(s)	52.96	30.36	5.04	1.16	8.04	364.76	5.92	12.24	1.24	7.40	1.36	39.64	4.84
	(b)	5.76	4.79	41.84	12.52	16.01	6.48	12.73	***10.11***	**158.55**	**9.49**	**27.89**	3.88	45.39
	av.	8.85	4.88	42.87	13.20	16.77	6.64	13.25	10.78	160.10	9.78	28.03	3.95	46.42
	std.	0.06	0.10	1.04	0.32	0.37	0.09	0.25	0.23	0.88	0.14	0.66	0.04	0.62
	t(s)	54.20	31.52	5.24	1.24	8.20	370.76	5.80	12.32	1.40	7.32	1.40	40.28	4.96
	(c)	***5.17***	4.82	42.77	12.55	16.42	***5.87***	**12.67**	10.46	160.29	9.58	28.58	***3.65***	**44.26**
	av.	5.38	4.97	43.07	12.92	16.81	6.25	13.26	10.69	162.79	9.71	28.71	3.73	45.92
	std.	0.09	0.08	0.58	0.41	0.20	0.21	0.21	0.11	1.29	0.13	0.47	0.11	0.81
	t(s)	104.36	60.20	6.20	1.24	29.28	1470.21	18.64	36.96	3.52	12.28	3.20	85.92	5.08
	(d)	***5.17***	**4.76**	**41.33**	12.84	***15.85***	6.02	13.01	10.41	160.29	9.57	28.37	***3.65***	44.55
	av.	5.37		42.71	13.23	16.52	6.38	13.32	10.70	162.79	9.74	28.42	3.73	45.91
	std.	0.08	0.09	0.63	0.28	0.33	0.23	0.13	0.11	1.29	0.12	0.45	0.11	0.97
	t(s)	114.12	65.4	6.68	1.40	33.16	1480.58	17.48	35.72	3.64	12.12	3.60	80.76	5.24

In the next set of experiments, the effect of incorporating the shuffling strategy using roulette wheel selection in the examination selection process is tested with n = 0, 3, 5, 10, 15 and $n = 3$ is chosen based on the statistical test that proved that size 3 is the best selection. As the results in Table 5.2 reveal, the addition of the boundary set strategy with roulette wheel selection performed better by providing eight better solutions compared with the swapping strategy with roulette wheel selection. The addition of the boundary set and the selection strategy performed the best with a combination of [SD-SD] while the best combination ordering for swapping with selection strategy is [LD-SD] and [SD-SD] where the equal number of best solutions is achieved. Comparing the average results obtained from the strategies without roulette wheel selection in Table 5.1 and the strategies with roulette wheel selection in Table 5.2, most of the time the incorporation of the shuffling strategy improves the performance of the approach.

From the perspective of the strategies, the swapping strategy of the boundary set with the difficult set indicates that the solution quality can be improved. However, this strategy produced higher standard deviation compared with the adding strategy and also generates a possibility of producing an infeasible solution during the search. With boundary size 3 and roulette wheel size 3, it is observed that the adding strategy obtained eight better results while the swapping strategy produced better results for only five problem instances.

5.2.3 Discussion on the Performance of the Algorithm on the Toronto Benchmark Datasets

The overall results once again highlight the importance of the methodology used to change the ordering of difficult examinations, particularly those causing infeasibility. In our approach, the ordering of the examinations within the difficult set with respect to the others appears to be vital, combined with the assignment strategy. As shown in Figure 5.7, the experiments on adding and swapping the boundary set and difficult set with a shuffling strategy of roulette wheel selection have resulted in the average number of the examinations in the difficult set varying with different ordering strategies. The approach using the largest degree ordering generates infeasibility more often for a given solution during the time-slot assignments compared with the one using the saturation degree ordering. This feature has contributed to the higher size of the difficult set. On the other hand, the saturation degree ordering might easily create a feasible solution for some problem instances (for example car91 and uta92 I). However, using saturation degree alone does not guarantee good quality of the solution. Adding or swapping the boundary set with the difficult set might increase the number of examinations in the difficult set. This has benefitted the shuffling strategy in attempting to avoid getting stuck during the search process.

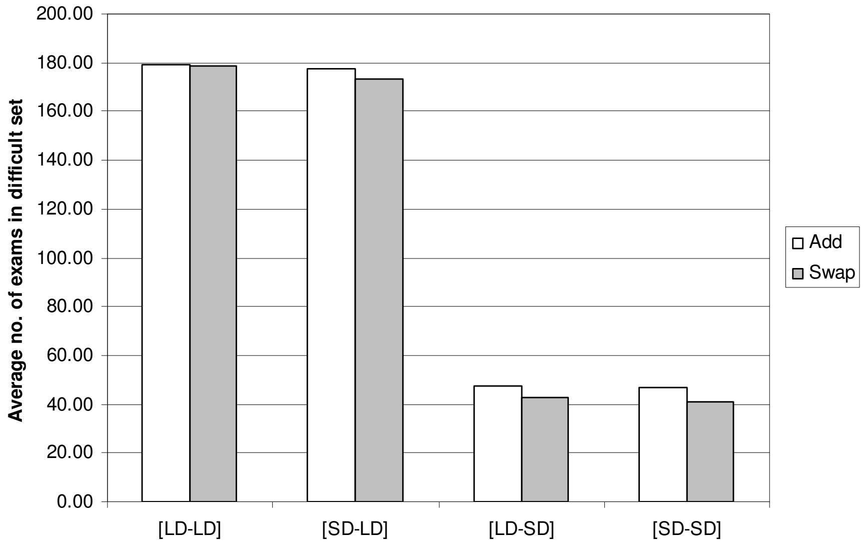


FIGURE 5.7: Average number of examinations in the difficult set (its size) over all problems considering all shuffling strategies using different initialisation and reordering heuristics of the Toronto benchmark datasets (Add = adding strategy, Swap = swapping strategy)

Figure 5.8 illustrates the size of the difficult set and the solution quality every 100 iterations for different combinations of initial ordering and reordering heuristics for the kfu93 problem instance. It shows that using the largest degree as initial ordering causes an increased number of examinations to generate infeasible solutions when compared with the saturation degree initial ordering. This infeasibility contributed to the size of the difficult set. The trend between the two dotted lines in the graphs shows that there is a significant drop in the solution quality when the size of the difficult set is increased for different heuristic combinations. The [SD-LD], however, does not demonstrate any improvement to the solution quality for a certain time, but only after the shuffling strategy of roulette wheel selection is incorporated. On the other hand, the [LD-SD] shows a slight movement and remains steady for a certain time even though there is a small increase in the number of examinations in the difficult set. Meanwhile, the [SD-SD] drastically changes the solution quality, which is significantly consistent with the increasing size of the difficult set. It is interesting to observe that the increasing size of the difficult set with [LD-LD] in this Figure 5.8 gives a higher possibility of achieving a good solution quality with the help of the boundary size and the shuffling strategy of roulette wheel selection.

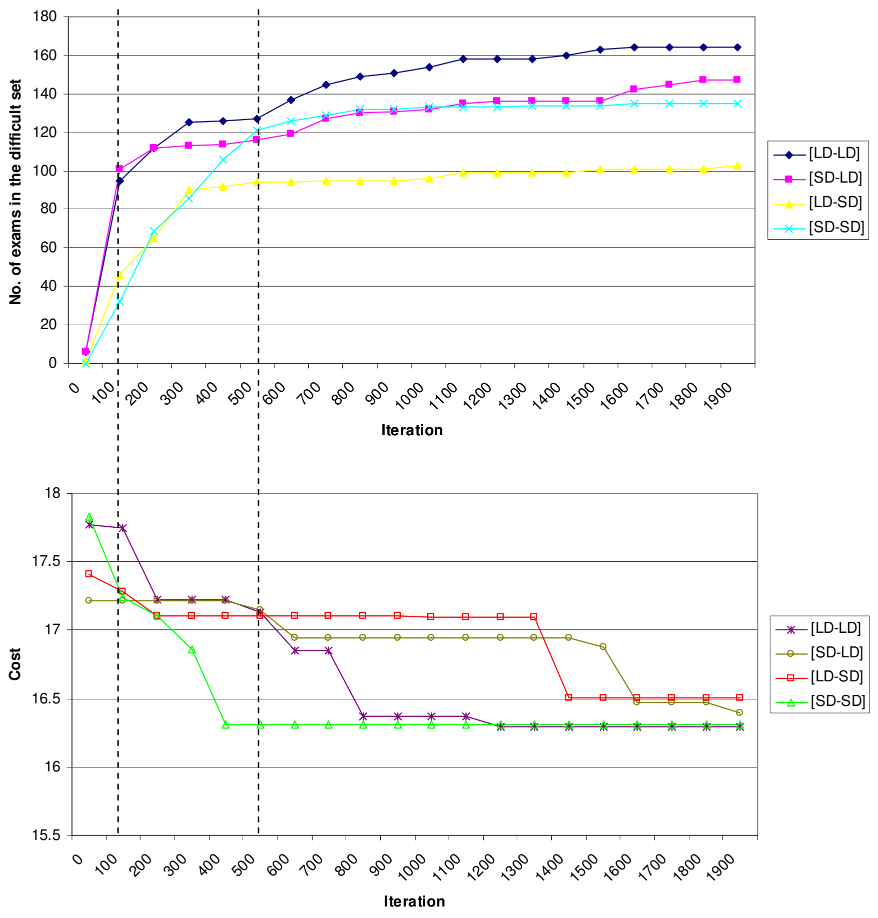


FIGURE 5.8: The change in the size of the difficult set and the solution quality at every 100 iterations during the sample runs for kfu93 (LD = largest degree, SD = saturation degree)

5.2.4 Comparison with the Previous Approaches on the Toronto Datasets

Tables 5.3 and 5.4 compare respectively the best results of the chosen heuristic combination and the best of all heuristics combination obtained from the strategy of roulette wheel selection with the previous results on constructive and improvement approaches. The result from the [SD-SD] of the adding strategy is chosen as it obtained the highest number of best solutions among other heuristic combinations from the adding strategy, for the sake of comparison with other constructive approaches. In Table 5.3, the results of [7] are from the [SD-SD] of the adding strategy and the results of [8] are the best results of all heuristic combinations, while Table 5.4 shows that [7] is the best results of the ADO approach.

The method of Qu and Burke (2007), as described in Section 5.1.4, is the closest comparison with the ADO approach as they also implemented a decomposition strategy. Comparing the solutions across all problem instances, it is observed that the ADO approach does not yield the best overall results on all problem instances within constructive approaches. On the other hand, it provides a better result when compared with the approach proposed by Qu and Burke (2007) for car91. Moreover, the ADO approach obtained better results than those reported by Carter et al. (1996) for four problems (car91, car92, sta83 I, tre92), Burke and Newall (2004) for one problem (sta83 I), Asmuni et al. (2009) for four problems (car91, rye93, sta83 I and ute92) and Burke et al. (2010e) for three problems (kfu93, sta83 I and ute92). However, Burke and Newall (2004) and Qu and Burke (2007) do not provide the result for rye93. Moreover, no other approaches provide results for pur93 I except for that of Carter and Laporte (1996). Burke and Newall (2004) did use the pur93 I instance, but they used a different variant of the instance than that tested in this study. For details on the constructive approaches, see Section 2.2.3.

Nevertheless, this approach does not performed well when compared with those in Qu and Burke (2007). Our approach keeps increasing the size of the difficult set from time to time whenever there is examination that cannot be scheduled into the timetable. However, we introduced the merging and swapping strategies in order to vary the ordering in the difficult set and this strategy has improved the solution quality but still the results are very far from Qu and Burke (2007) except for car91. We assumed that the strategy to increase the difficult set size without considering releasing some other examinations in the difficult set might be the reason why the results are not so good as Qu and Burke (2007). It is suggested in the future study that some examinations in the difficult set should be released and this would involve some analysis on identifying which examination should be released and should be considered no longer difficult to be scheduled. On the other hand, this study found that at a certain point, the size of the difficult set might be settled and this has helped in better ordering when using the roulette wheel selection strategy where this ordering combines the information from the largest degree and saturation degree value. The examinations in the boundary set also help in better ordering where these examinations are coming from examination in the easy set i.e. these examinations have higher difficulty value within the easy set. They are either merged or swapped with the examination in the difficulty set so that a different ordering could be obtained after the process.

The results of the ADO approach are also compared with those obtained using other improvement approaches which have incorporated a multi-phase processing that involves the construction of an initial solution before proceeding with the improvement of the solution quality. Table 5.4 shows the comparison of the improvement approaches with the ADO approach. The results depict clearly that the ADO approach is broadly comparable with the improvement strategies. The results of the ADO approach are better than those of Di Gaspero and Schaerf (2001) for nine problem instances (car91, car92, hec92

Table 5.3: Comparison of different constructive approaches of the Toronto benchmark datasets (The bold entries indicate the best results for constructive approaches only, while those in italic and bold indicate the best results for the decomposition approach)

Problem	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
car91	7.10	**4.97**	5.29	5.08	5.03	5.45	***5.31***	5.17
car92	6.20	**4.32**	4.54	4.38	4.22	***4.5***	4.76	4.74
ear83 I	36.40	36.16	37.02	38.44	36.06	**36.15**	40.91	40.91
hec92 I	**10.80**	11.61	11.78	11.61	11.71	***11.38***	12.36	12.26
kfu93	**14.00**	15.02	15.80	14.67	16.02	***14.74***	16.31	15.85
pur93 I	3.90	-	-	-	-	-	***6.07***	5.87
lse91	**10.50**	10.96	12.09	11.69	11.15	***10.85***	12.84	12.58
rye93	**7.30**	-	10.38	9.49	9.42	-	***10.40***	10.11
sta83 I	161.50	161.90	160.40	157.72	158.86	**157.21**	159.20	158.12
tre92	9.60	**8.38**	8.67	8.78	8.37	***8.79***	9.30	9.30
ute92	**25.80**	27.41	28.07	26.63	27.99	***26.68***	27.80	27.71
uta92 I	3.50	**3.36**	3.57	3.55	3.37	***3.55***	3.74	3.65
yor83 I	41.70	40.88	**39.80**	40.45	39.53	***42.2***	43.98	43.98

[1] Carter et al. (1996), [2] Burke and Newall (2004), [3] Asmuni et al. (2009), [4] Abdul Rahman et al. (2009), [5] Burke et al. (2010e), [6] Qu and Burke (2007), [7] ADO with [SD-SD] and RWS, [8] Best of ADO for $\delta = 3$ and $n = 3$)

I, kfu93, lse91, sta83 I, tre92, ute92, uta92 I), Paquete and Fortseca (2001) for three problem instances (kfu93, lse91, ute92) and a tie with tre92, Burke and Newall (2003) for (sta83 I), Yang and Petrovic (2004) for (sta83 I), Abdullah et al. (2007) for (car91, sta83 I), Eley (2007) for one problem instance (car91), Caramia et al. (2008) for four problem instances (car91, car92, sta83 I, tre92) and Turabieh and Abdullah (2011) for one problem instance (sta83 I). Only three of the approaches (Eley (2007), Caramia et al. (2008) and Turabieh and Abdullah (2011)) provided a result for pur93 I.

5.2.5 Implementation on the ITC2007 benchmark datasets

The proposed approach is also tested on the ITC2007 benchmark datasets. As with the previous test, the roulette wheel selection strategy made a significant improvement to the solution quality to the Toronto benchmark datasets. In this case, the test on the ITC2007 benchmark datasets is implemented with the roulette wheel selection strategy. The same parameter setting as in the Toronto benchmark datasets is employed where the boundary set size and the roulette wheel selection size are 3. Figure 5.9 illustrates the average number of examinations in the difficult set based on the outcome from the tested experiment. It shows that the largest degree ordering generates more infeasible examinations compared with the saturation degree ordering. On the other hand, the saturation degree ordering performed better in terms of generating a feasible solution

TABLE 5.4: Comparison of different improvement approaches (The bold entries indicate the best results)

Problem	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	ADO
car91	6.2	-	4.65	5.1	**4.5**	5.4	5.2	5.2	6.6	4.6	4.8	5.17
car92	5.2	-	4.1	4.3	3.93	4.2	4.4	4.3	6	**3.9**	4.1	4.74
ears83 I	45.7	38.9	37.05	35.1	33.71	34.2	34.9	36.8	**29.3**	32.8	34.92	40.91
hec92 I	12.4	11.2	11.54	10.6	10.83	10.4	10.3	11.1	**9.2**	10	10.73	12.26
kfu93	18	16.5	13.9	13.5	13.82	14.3	13.5	14.5	13.8	**13**	**13**	15.85
lse91	15.5	13.2	10.82	10.5	10.35	11.3	10.2	11.3	**9.6**	10	10.01	12.58
pur93 I	-	-	-	-	-	-	-	4.6	**3.7**	-	4.73	5.87
rye92	-	-	-	8.4	8.53	8.8	8.7	9.8	**6.8**	-	9.65	10.11
sta83 I	160.8	158.1	168.73	157.3	158.35	157.0	159.2	157.3	158.2	**156.9**	158.26	158.12
tre92	10	9.3	8.35	8.4	7.92	8.6	8.4	8.6	9.4	7.9	**7.88**	9.30
uta92 I	4.2	-	3.2	3.5	**3.14**	3.2	3.6	3.5	3.5	3.2	3.2	3.65
ute92	29	27.8	25.83	25.1	25.39	25.3	26	26.4	**24.4**	24.8	26.11	27.71
yor83 I	41	38.9	37.28	37.4	36.53	36.4	36.2	39.3	36.2	**34.9**	36.22	43.98

[1] Di Gaspero and Schaerf (2001), [2] Paquete and Fortseca (2001), [3] Burke and Newall (2003), [4] Merlot et al. (2003), [5] Yang and Petrovic (2004), [6] Côté et al. (2005), [7] Abdullah et al. (2007), [8] Eley (2007), [9] Caramia et al. (2008), [10] Burke et al. (2010a), [11] Turabieh and Abdullah (2011) and [12] Best of ADO for $\delta = 3$ and $n = 3$

than the largest degree ordering. This shows the same behaviour as the Toronto benchmark datasets on the contribution of the number of infeasible examinations to different heuristic orderings. Nevertheless, the difference between these two types of ordering is not as great as in the Toronto benchmark datasets.

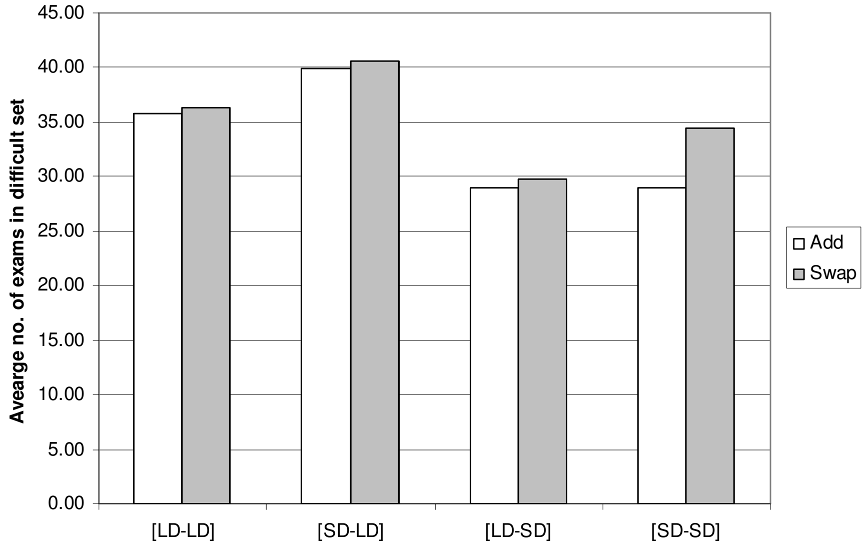


FIGURE 5.9: Average number of examinations in the difficult set (its size) over all problems considering all shuffling strategies using different initialisation and reordering heuristics of the ITC2007 benchmark datasets (Add = adding strategy, Swap = swapping strategy)

Figure 5.10 illustrates the size of the difficult set and the solution quality every 100 iterations for different combinations of initial ordering and reordering heuristics for Exam_4 of the ITC2007 benchmark datasets tested with the ADO approach with roulette wheel selection strategy. As can be seen, the analysis between the two dotted lines shows that the saturation degree initial ordering creates more unscheduled examinations for Exam_4 compared with the largest degree initial ordering. The number of unscheduled examinations for each heuristic combination shows an upward movement throughout the iterations. On the other hand, the solution quality demonstrates a downward movement for each heuristic combination. Nevertheless, the solution quality shows a drop-off at the first 200 iterations while for the remaining iterations the solution quality remains steady. With the exception of [SD-SD], the solution quality shows a decrement throughout the iterations. However, this heuristic combination indicates the worst of the solution quality. It is assumed that this occurred because of the nature of the problem, where Exam_4 is one of the difficult problem instances among the ITC2007 benchmark datasets with a conflict density of 0.15. Moreover, the problem has a difficult hard constraint requirement to cater for scheduling forty hard constraint time-slots.

Table 5.5 shows the solution quality obtained using the parameter setting with the adding and the swapping strategy. It is interesting to note that the swapping strategy achieved

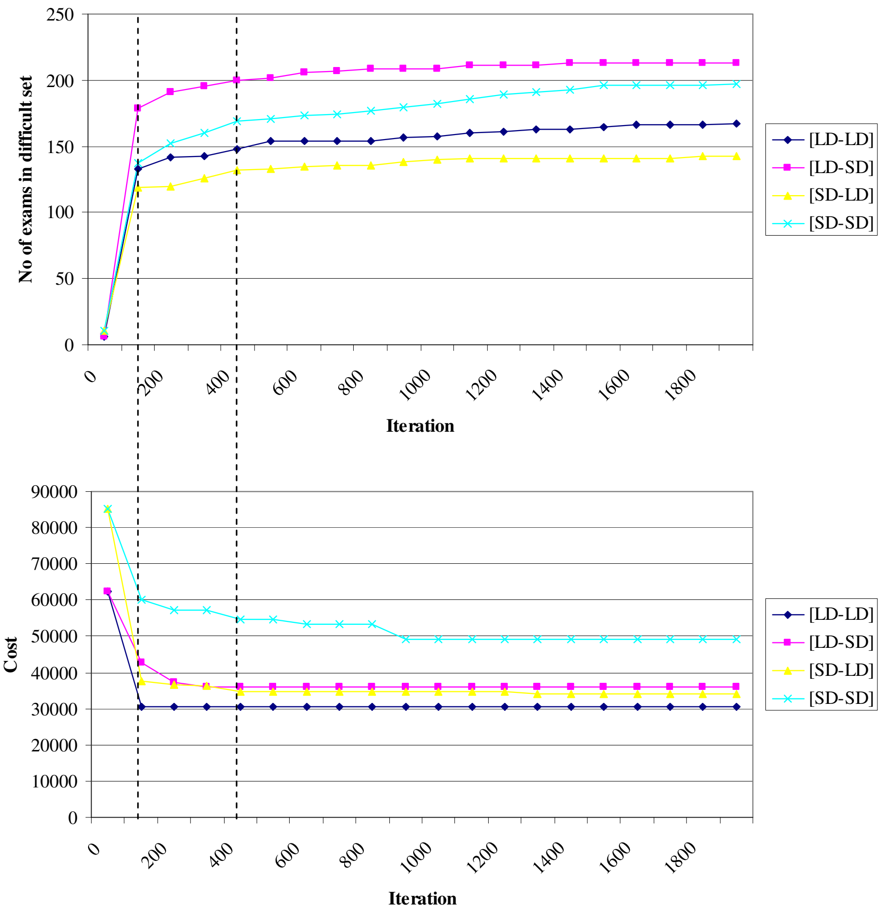


FIGURE 5.10: The change in the size of the difficult set and the solution quality every 100 iterations during the sample runs for Exam_4 of the ITC2007 benchmark datasets (LD = largest degree, SD = saturation degree)

ten out of twelve better results compared with the adding strategy. This obviously differs from the results obtained from the experiments from the Toronto benchmark datasets, when the adding strategy performed better than the swapping strategy. This may be due to the nature of the ITC2007 benchmark problems that includes consideration of room capacity and other hard and soft constraints. Considering the best heuristic combination within this strategy, it can be observed that the adding strategy performed effectively with the saturation degree ordering of the difficult set, with eight problem instances achieving better results. On the other hand, the performance for both largest degree and saturation degree ordering for the difficult set is equal in terms of the number of best results obtained.

A comparison of different approaches for the ITC2007 benchmark datasets with the ADO approach is illustrated in Table 6.8. It can be seen that the ADO approach is not close to other approaches within the ITC2007 benchmark datasets. This could be because of the ADO approach is simply a constructive approach, while all other approaches employed a multi-phase strategy in order to improve the solution quality. It is clear

Table 5.5: Comparing solution quality of the ITC2007 benchmark datasets for (a) [LD-LD], (b) [LD-SD], (c) [SD-LD] and (d) [SD-SD] with shuffling strategies of adding the boundary set into the difficult set and swapping examinations between the boundary and difficult sets with $\delta = 3$ and includes roulette wheel selection for examinations with $n = 3$ (av.=average solution quality; std.=standard deviation) (Bold font indicates the best for different ordering and strategy and bold and italic is the best of all for each problem instance)

		E_1	E_2	E_3 I	E_4	E_5	E_6	E_7	E_8	E_9	E_10	E_11	E_12
Add the boundary set ($\delta = 3$) into difficult set	(a)	11498	3330	***20403***	26987	8801	28935	16577	20725	**2272**	***16876***	***46649***	10881
	av.	11876.0	3611.9	21510.7	31284.3	9199.7	29565.5	17374.0	21186.7	2413.52	17633.3	52210.7	16053.8
	std.	162.1	101.5	361.6	2196.9	171.6	309.4	324.4	220.1	47.5	286.8	1920.2	1435.7
	(b)	***11378***	3400	20405	40791	8794	***28745***	16610	**20508**	2273	16938	47967	11747
	av.	11928.5	3604.0	21467.1	52524.4	9152.6	29584.5	17352.6	21217.7	2412.1	17642.4	51314.9	14941.8
	std.	184.6	91.9	417.9	6290.6	166.7	321.4	250.3	294.0	42.8	241.4	1623.4	1843.0
	(c)	11417	3379	20719	***25449***	**8226**	28815	16734	21092	2290	17555	47520	11233
	av.	12028.9	3625.1	21456.0	30142.6	8815.7	29472.0	17451.1	21662.8	2424.0	18127.8	51146.4	14172.9
	std.	155.5	119.0	376.4	2435.2	165.6	252.0	316.7	275.6	44.0	245.0	1806.8	1807.6
	(d)	11645	***3243***	20457	42931	8272	29090	**16403**	20927	2307	17383	50331	***7445***
	av.	12006.5	3602.2	21500.9	59745.9	8833.0	29889.0	17402.0	21652.1	2424.5	18130.2	59891.9	15138.5
	std.	177.8	109.2	430.8	6239.9	189.6	366.1	324.0	336.8	38.3	262.3	2759.7	2227.6
Swap in the boundary ($\delta = 3$) and difficult set	(a)	11438	**3245**	**20559**	27902	8551	**28750**	16547	20559	2330	17319	48606	10803
	av.	11893.4	3596.2	21454.2	32801.8	9175.1	29689.8	17391.6	21293.4	2419.1	17708.0	53848.0	14225.8
	std.	146.1	96.3	397.4	2106.2	171.1	417.6	297.2	331.4	35.6	214.9	2480.7	1963.1
	(b)	11652	3342	20606	39238	8903	29300	16497	20721	2277	**16946**	47814	**10306**
	av.	11921.6	3615.9	21414.5	48991.1	9213.2	29805.9	17281.0	21274.2	2415.5	17674.5	51841.7	15496.6
	std.	142.2	107.9	357.4	5651.6	159.2	344.2	270.2	273.9	51.5	246.0	1833.6	1798.7
	(c)	11421	3425	20672	**27035**	***8199***	28950	16699	***20308***	2248	17440	48563	11156
	av.	11960.4	3650.4	21505.7	31811.5	8844.6	29412.9	17503.0	21631.1	2414.5	18188.0	51639.6	14672.5
	std.	209.0	87.7	402.8	2072.8	195.4	236.9	297.5	382.2	49.5	234.6	1741.3	1917.6
	(d)	**11406**	3477	20711	32096	8482	28785	***16381***	20587	***2171***	17571	**46804**	11311
	av.	11952.7	3641.0	21530.3	45996.6	8824.2	29462.2	17364.7	21634.6	2418.1	18193.3	50494.0	15828.8
	std.	198.3	82.7	322.3	5957.8	151.3	256.5	334.3	319.9	57.6	249.4	1647.5	1482.4

that a modification is essential for incorporating other improvement methods in order to improve the performance of the ADO approach. A discussion of the improvement approach is provided in Chapter 6.

5.3 Conclusion

This study discusses an approach based on adaptive strategies that decomposes the examinations in a given problem into two sets: difficult to schedule examinations and easy to schedule examinations. This decomposition is performed automatically at each iteration, and is augmented with a suitable ordering of examinations within each set. In this study, it is observed that by merging or swapping the boundary set with the difficult set, solution quality could be improved. A stochastic component based on roulette wheel selection is embedded into the approach in order to shuffle the order of examinations. This mechanism provides a higher chance that an examination with a higher score will be selected for timetabling. Different parameters are tested on the boundary size and roulette wheel selection size and the parameter setting is undertaken based on the statistical analysis. It is observed that, using saturation degree heuristic, the possibility of creating infeasible solutions could be decreased and that dynamic ordering achieves better ordering of examinations in the list. This study shows that the proposed approach is simple to implement, yet it is competitive with previously published constructive and improvement approaches. In this study, the same ordering heuristics are used for reordering the examinations in the difficult and easy sets. The proposed framework allows the use of different strategies. The next chapter will discuss the improvement strategy for enhancing solutions obtained from the proposed constructive approaches.

TABLE 5.6: Comparison of different approaches for the ITC2007 benchmark datasets (The bold entries indicate the best results)

Problem	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	ADO
Exam_1	4370	5905	8006	6670	12035	4370	4633	4699	8559	6235	4775	**4128**	4368	11378
Exam_2	400	1008	3470	623	3074	385	405	385	830	2974	385	**380**	390	3243
Exam_3	10049	13862	18622	-	15917	9378	9064	8500	11576	15832	8996	**7769**	9830	20403
Exam_4	18141	18674	22559	-	23582	15368	15663	14879	21901	35106	16204	**13103**	17251	25449
Exam_5	2988	4139	4714	3847	6860	2988	3042	2795	3969	4873	2929	**2513**	3022	8199
Exam_6	26950	27640	29155	27815	32250	26365	25880	25410	28340	31756	25740	**25330**	25995	28745
Exam_7	4213	6683	10473	5420	17666	4138	4037	3884	8167	11562	4087	**3537**	4067	16381
Exam_8	7861	10521	14317	-	16184	7516	7461	7440	12658	20994	7777	**7087**	7519	20308
Exam_9	1047	1159	1737	1288	2055	1014	1071	-	-	-	964	**913**	-	2171
Exam_10	16682	-	15085	14778	17724	14555	14374	-	-	-	13203	**13053**	-	16876
Exam_11	34129	43888	-	-	40535	31425	29180	-	-	-	28704	**24369**	-	46649
Exam_12	5535	-	5264	-	6310	5357	5693	-	-	-	5197	**5095**	-	7445

[1] Müller (2008), [2] Gogos et al. (2008), [3] Atsuta et al. (2008), [4] De Smet (2008), [5] Pillay (2008), [6] Müller (2009), [7] McCollum et al. (2009), [8] Gogos et al. (2009), [9] Pillay (2010a), [10] Burke et al. (2010f), [11] Gogos et al. (2010a), [12] Gogos et al. (2010b), [13] Turabieh and Abdullah (2011))

Chapter 6

A Variable Neighbourhood Search - Great Deluge for Examination Timetabling Problem

Most successful approaches to examination timetabling, including that adopted by the winner of the ITC2007 competition (Müller, 2009), consist of multiple stages, in which a constructive approach is used in finding a good initial solution, and then one or more improvement approaches are employed to further improve the quality of solution obtained in the previous stage. The main objectives of this chapter are to show how previously constructed solutions are improved and to investigate the influence of various initialisation methods and neighbourhood orderings on the performance of the search algorithm. The chapter presents a variable neighbourhood search approach combined with a great deluge acceptance method, allowing the acceptance of some worsening solutions for solving the examination timetabling problem. A range of neighbourhood structures were tested within this approach for diversification purposes. The neighbourhood orderings in VNS and the effect of the initialisation methods on the solution quality of the improvement approach were investigated in relation to two well-known examination timetabling benchmarks. The results illustrate the success of the variable neighbourhood search - great deluge (VNS-GD) approach in solving examination timetabling problems. This study has shown that initialisation is crucial for the success of the improvement approach used in a multistage setting, particularly for VNS. The present chapter describes variable neighbourhood search in Section 6.1. The details of the variable neighbourhood search - great deluge approach are discussed in Section 6.2, focusing on the algorithmic components, initialisations, neighbourhood structures and acceptance criteria. Section 6.3 provides the experimental results and comments. Finally, the conclusion is presented in Section 6.4.

6.1 Variable Neighbourhood Search (VNS)

Most local search approaches could trap at a local minimum during the search process. As described by Mladenović and Hansen (1997) and Hansen and Mladenović (2001), different neighbourhood structures have their own local minimum. Motivated by this, the use of a set of neighbourhood structures was introduced by Mladenović and Hansen (1997) so that the search process could continue by changing to one of the other neighbourhood structures, while at the same time the search could avoid being trapped at a local minimum. VNS is a descent-ascent approach that iteratively applies a shaking strategy and local search in order to find the best solution for a problem at hand. This algorithm can be considered as a single point-based selection hyper-heuristic (Burke et al., 2010b) which requires several neighbourhood structures with different natures. If one of these fails to improve the solution, the other may still have a chance. The VNS main framework consists of three main steps: 'shaking', 'local search' and 'move'.

Algorithm 13 below illustrates the steps of the basic VNS algorithm adopted by Mladenović and Hansen (1997). N_k is a set of neighbourhood structures that will be selected during the search, where $k = 1, ..., k_{max}$. The foundation of basic VNS was a descent approach that only accepts the improving move. The current solution is accepted if it is better than the incumbent solution. In this circumstance, the neighbourhood structures are alternated in order to avoid local optimum since different neighbourhood structures have their own local optimum. The local search procedure acts as an intensification strategy to converge to a good solution. At the shaking procedure, the point s' is generated randomly. The shaking procedure is a diversification strategy that avoids cycling during the search. There are several stopping conditions utilised in this approach, such as the maximum number of iterations, the number of non-improving iterations and the preset of CPU time.

Algorithm 13 Basic VNS algorithm

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

The VNS approach has been successfully implemented in the field of university timetabling. Wong et al. (2005), for example, implemented variable neighbourhood descent using multiple neighbourhood structures for solving the un-capacitated problem of examination timetabling. Each neighbourhood structure was formulated into a different local search operator in order to explore and exploit the search space of solutions. The aim was to balance the intensification and diversification during the exploration of the search space. Furthermore, Ahmadi et al. (2003) employed the VNS as a high-level heuristic that chose low-level heuristics. Their study described the application of VNS as a perturbation-based algorithm to the examination timetabling problem that incorporated weight values to each low-level heuristic in order to find good quality solutions. The low-level heuristics used within this hyper-heuristic were based on a combined selection of an examination, time-slot and room.

A recent study has incorporated a genetic algorithm as neighbourhood selector within a VNS approach. Burke et al. (2010a) showed that VNS and hybridisation with a genetic algorithm could yield good quality solutions, a method which produced several best known results in the literature. In their study, the genetic algorithm imitated the concept of hyper-heuristics and case-based reasoning, where it was not directly applied to the problem but instead worked at a high-level. Since the solution quality was dependent on the selection of a neighbourhood, the genetic algorithm performed the search by selecting the list of neighbourhoods from the VNS framework.

The VNS approach was implemented for the course timetabling by Abdullah et al. (2005). The course timetabling problem required a search for the best assignment of lectures to time-slots and rooms, subject to constraints. The study employed a VNS approach using exponential Monte-Carlo as an acceptance criteria for worsening moves, and during the search, a tabu list was utilised to prohibit a non-improving neighbourhood being used for a certain period.

There have been further recent studies based on the VNS approach. For example, VNS was employed to solve a nurse rostering problem (Burke et al., 2003b), a graph colouring problem (Avanthay et al., 2003), a median cycle problem (Morena Pérez et al., 2003), project scheduling (Fleszar and Hindi, 2004) and an external graph problem (Caporossi and Hansen, 2000).

6.2 VNS for Examination Timetabling

Studies by Hansen and Mladenović (2001), Burke et al. (2003b) and Abdullah et al. (2005) have shown that the choice of neighbourhood structures and their ordering (changing from one to another) within the VNS framework could considerably influence the solution quality. The neighbourhood structures were ordered on the basis of a pre-defined sequence with increasing step size, on the assumption that this might be the best sequence of neighbourhood structures yielding a better solution quality. At the same time, the running time could be reduced since the search always started with a small neighbourhood structure each time an improvement occurred. Burke et al. (2003b), introduced a parameter $success_k$ to penalise a non-improving neighbourhood structure. In the case that an improvement to the solution quality was found, the search always started with the first neighbourhood with the smallest size, while at the same time considering the $success_k$ value for choosing a neighbourhood. Any neighbourhood that had $success_k$ less than 1, could not have priority in the next iteration as it showed that the neighbourhood could not make any improvement to the solution quality. Nevertheless, a recent study by Burke et al. (2010a) has shown that the ordering of neighbourhood structure was not essential: the neighbourhood structure was chosen by a genetic algorithm and the solution quality depended on the selection of a neighbourhood.

Motivated by these studies, an investigation has been undertaken into the ordering of neighbourhood structures, in which five variants of neighbourhood ordering are explored. The first and second variants are the ordering of neighbourhood structures based on increasing size, as implemented by Burke et al. (2003b) and Abdullah et al. (2005): the first variant is a basic VNS (as illustrated in Algorithm 13) where the next neighbourhood structure to be used always starts with $k = 1$ whenever an improvement is found; the second variant is when each time improvement is found, the current neighbourhood k is used in the next iterations for further search. These variants of neighbourhood orderings are represented by 'basic VNS' and 'start-k'.

The third variant of neighbourhood ordering is based on the strategy adapted from the squeaky wheel optimisation (Joslin and Clements, 1999), by giving penalty to the parameter 'priority' to the non-improving moves. If there is no improvement to the solution quality when using the current k neighbourhood, then the next neighbourhood to be chosen is the one with the lowest priority value. As illustrated in Algorithm 14, a parameter, $priority_k$, is increased to any neighbourhood that cannot improve the current solution. This value is increased by one each time an improvement to the solution quality cannot be found. In this study, this ordering strategy is represented by 'adaptive I'. The effect of small size neighbourhood on the adaptive change of neighbourhood structure is also investigated: the fourth variant of neighbourhood is similar to the third but gives greater priority to a small neighbourhood to be chosen first (it is referred as 'adaptive II'). In the case where more than one neighbourhood has the same priority value, then the smaller size neighbourhood is chosen first in the next iteration.

In this circumstance, the neighbourhood structures are first classified into small and large group sizes, based on the effect on the chosen moves. For example, the small neighbourhood structure consists of the single move of an examination to a new feasible time-slot, while the large neighbourhood consists of a number of examinations to be moved and might incur large differences to the solution quality when the chosen move takes place, for instance, when two time-slots are swapped at random. This neighbourhood structure involves all examinations in one time-slot to be swapped with all examinations in the other time-slot.

Algorithm 14 Adaptive VNS algorithm

```
Select the set of neighbourhood structures $N_k, k = 1, ..., k_{max}$, to be used in the search;
set initial solution $s$; choose stopping condition;
Repeat until stopping criteria is satisfied:
1. Set $k := 1$;
2. Until $k = k_{max}$, repeat:
(a) Shaking: Generate a point $s'$ at random from the $k^{th}$ neighbourhood of $s(s' \in N_k(s))$;
(b) Local Search: Apply a local search method with $s'$ as initial solution until local optimum $s''$ is obtained.
(c) Move or not:
if $s''$ is better than incumbent solution or is accepted based on acceptance criterion then
  $s \leftarrow s''$
else
  $priority_k \leftarrow priority_k + 1$
end if
Continue the search with $k$ that has the lowest priority value and if more than one neighbourhood that has highest priority then choose $k$ that is from small size of neighbourhood.
```

As the fifth variant of neighbourhood ordering, the reinforcement learning mechanism is also investigated, a strategy which has previously been implemented and discussed in Section 4.1.4.3 of Chapter 4. Reinforcement learning is a mechanism that interacts with the behaviour of an environment by assigning punishments and rewards based on its performance. In this case, the improving move of a neighbourhood is given a reward, while a punishment is given if the move fails to improve the current solution. Each time the reward or punishment is assigned, the score value $score_k$ of a k neighbourhood is increased or decreased by one. Initially, the $score_k$ is initiated with 5. The increment of $score_k$ value never exceeds the value of ten and remains constant if the neighbourhood continues to improve, while the decrement never exceeds the value of zero and remains constant if the neighbourhood keeps failing to improve. This variant of neighbourhood ordering in this study is referred as ‘RL’. The pseudo-code of the reinforcement learning algorithm is illustrated in Algorithm 15.

Algorithm 15 Reinforcement learning VNS algorithm

```
Select the set of neighbourhood structures $N_k, k = 1, ..., k_{max}$, to be used in the search;
set initial solution $s$; choose stopping condition;
Repeat until stopping criteria is satisfied:
1. Set $k := 1$;
2. Until $k = k_{max}$, repeat:
(a) Shaking: Generate a point $s'$ at random from the $k^{th}$ neighbourhood of $s(s' \in N_k(s))$;
(b) Local Search: Apply a local search method with $s'$ as initial solution until local optimum $s''$ is obtained.
(c) Move or not:
if $s''$ is better than incumbent solution or is accepted based on acceptance criterion then
  $s \leftarrow s''$
  $score_k \leftarrow score_k + 1$
  if $score_k > 10$ then
    $score_k = 10$
  end if
else
  $score_k \leftarrow score_k - 1$
  if $score_k < 0$ then
    $score_k = 0$
  end if
end if
Continue the search with $k$ that has highest score and if more than one neighbourhood has highest score then choose $k$ randomly.
```

6.2.1 Initialisation

The study is concerned with improving the initial solution obtained from the constructive approaches in previous chapters. The main objective is to understand the influence of different kinds of initialisation to the solution quality when an improvement approach is employed. Several studies have shown that good initial solutions may ultimately translate to a further better quality solution within a short time (Burke and Newall, 2003, Gogos et al., 2010a). Three different variants of initial solutions are considered, i.e. poor, good and multiple. The good initial solution is the best obtained using the previous constructive approaches, while the poor initial solution is that which is worse by at least 20% compared with the good initial solution. The good and poor initial solutions are only one value and they are kept and used for a number of runs for improvement, while the multiple initial solutions are generated at each run before proceeding with the improvement approach, and their quality is varied. It is possible that some of them are not feasible, although this infeasibility has occurred only in relation to the ITC2007 benchmark datasets, and can be dealt with during the improvement phase with a repair mechanism that is incorporated into the VNS-GD algorithm. The multiple initial solutions are obtained from the constructive approach of Adaptive Heuristic Orderings described in Chapter 3.

6.2.2 Neighbourhood Structures

The choice of neighbourhood structure affects the search for better solution quality. The purpose of employing more than one neighbourhood structure is to attempt to ensure that it is possible to escape from a local optimum. This is due to the fact that each neighbourhood structure tends to have a different local minimum. In this case, if one neighbourhood structure fails to improve the current solution, then the other neighbourhood structures might still have a chance. Recently, the Kempe-chain move has been successfully applied to timetabling problems, presented in studies by Casey and Thompson (2003), Merlot et al. (2003), Côté et al. (2005), Tuga et al. (2007), Shaker and Abdullah (2009), Burke and Bykov (2006), Burke et al. (2010a), Gogos et al. (2010b) and Abdullah et al. (2010). The first implementation of the Kempe-chain move in timetabling was by Thompson and Dowsland [1996b], the focus of their study being to investigate the robustness of a simulated annealing approach with varying cooling schedules and three different neighbourhood structures in relation to the examination timetabling problem: the standard, the Kempe-chain and the *s*-chain. It was found that the Kempe-chain neighbourhood structure outperformed the others, and it was therefore concluded that the neighbourhood selection in the simulated annealing approach contributed significantly to a better quality timetable.

A standard neighbourhood structure as discussed by Thompson and Dowsland (1996b) is a single move neighbourhood which chooses an examination randomly and moves it to a new feasible time-slot. In addition to this simple neighbourhood structure, Kempe-chain is introduced as a variant. The Kempe-chain neighbourhood operates over two subsets of examinations by swapping between two feasible time-slots. Each subset of examinations is connected by edges to represent conflict between the examinations. Figures 6.1 (a) and 6.1 (b) show the standard Kempe-chain before and after the move. For instance, Figure 6.1 (a) depicts how e1 is chosen to be moved to a new time-slot, t2. In this circumstance, the Kempe-chain of e1 contains {e1, e2, e3, e5, e6, e8}, shaded in colour. Let us suppose e1 has to be moved to t2. However, the single move is impossible since conflict occurs with e5, e6 and e7. In this case, all examinations that are connected to e1 are swapped between these two time-slots, as shown in Figure 6.1 (b). On the other hand, any other examinations that are not connected to e1 remain in the current time-slot. This variant of Kempe-chain is called pair-wise Kempe-chain, a term used by Tuga et al. (2007), while a study by Thompson and Dowsland (1996b) named this

Kempe-chain as s-chain. The pair-wise Kempe-chain consists of a k-pair of Kempe-chain in a set of neighbourhood structures, where k is a positive integer value. In this circumstance, Figure 6.1 is an example of a one-pair Kempe-chain or one-chain ($s = 1$).

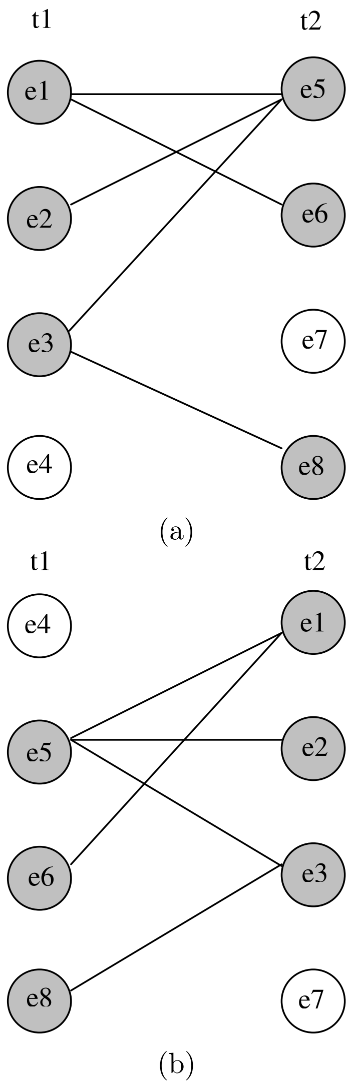


FIGURE 6.1: The one-pair Kempe-chain (a) before and (b) after the move

As the most common Kempe-chain neighbourhood in examination timetabling considers only a single move between two distinct time-slots, another variant of Kempe-chain is a two-pair Kempe-chain that involves examinations connected within k different time-slots. The Kempe-chain described in Thompson and Dowsland (1996b) used $s = 2$, while Tuga et al. (2007) chose k number of pairs randomly. Figures 6.2 (a) and 6.2 (b) below show an example of a two-pair Kempe-chain. For instance, e1 is chosen to be moved to a new time-slot. As Figure 6.2 (a) illustrates, the Kempe-chain of e1 contains {e1, e2, e3, e5, e6, e8, e9, e10, e11}, connected with edges and all connected examinations shaded in colour. The intention is to swap between two distinct time-slots - in this case, all the connected examinations in t1 and t3 are swapped. Figure 6.2 (b) shows that the connected examinations remain in the current time-slot, t2.

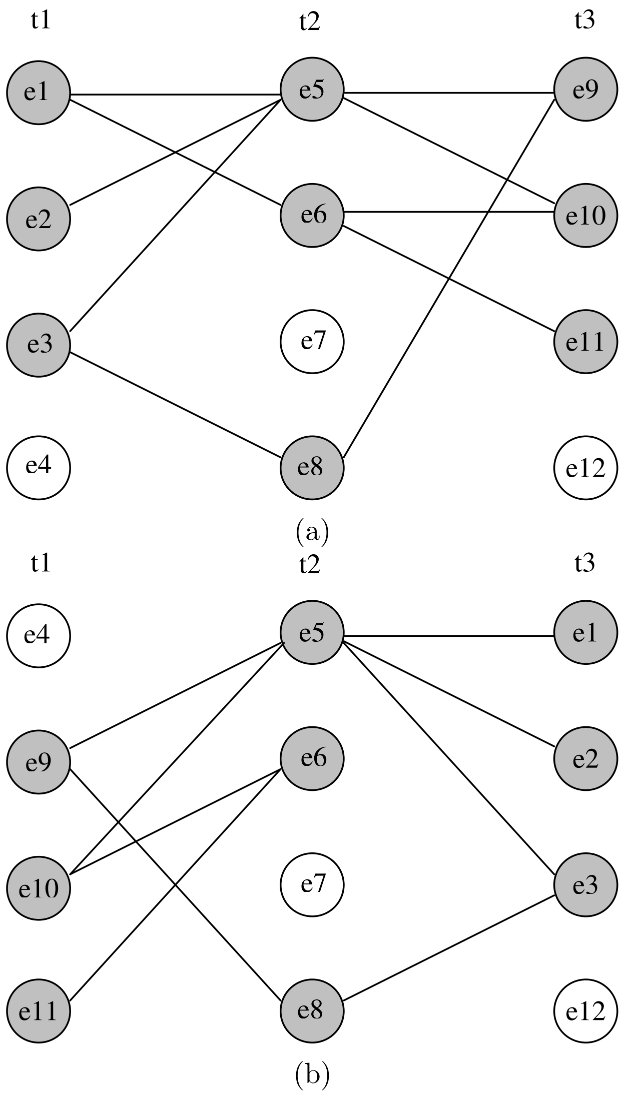


FIGURE 6.2: The two-pair Kempe-chain (a) before and (b) after the move

It was decided to use only the one-pair Kempe-chain because the two-pair chain did not perform well when compared with the one-pair chain, based on the initial test to the proposed approach. It was also proven by Thompson and Dowsland (1996b) that the one-pair Kempe-chain was the best neighbourhood to be used when compared with the s-chain neighbourhood with $s = 2$.

The neighbourhood structures in this study allow for infeasible moves. In the case of the Kempe-chain neighbourhood, the infeasible move may be due to more examinations from different time-slots being connected to the chosen examination. One of the aims of multiple neighbourhood structures in VNS is to shake the current solution s in various ways. In this case, the neighbourhood structures shake the solution s by allowing an infeasible move. In order to treat this infeasibility, a repair mechanism is invoked to the VNS algorithm. The detail of this process is discussed in Section 6.2.3.

Fifteen neighbourhood structures are considered in this study. As stated earlier, in order to generate the best sequence of neighbourhood structures, these are initially ordered based on their increasing size. Biased neighbourhood structures as used in Burke et al. (2010a) and Abdullah et al. (2005) were also considered. These neighbourhood structures (neighbourhood 8, 9, 11 and 12) are additionally considered - they choose the highest penalty value examination from a number of examinations that are selected randomly. The implemented neighbourhood structures are ordered as follows:

1. One examination at random and move to a new random feasible time-slot.
2. Two examinations at random and move each examination to a new random feasible time-slot.
3. Two examinations at random and swap the time-slots between these two examinations. The feasibility of the two examinations is maintained.
4. Three examinations at random and move each examination to a new random feasible time-slot.
5. Four examinations at random and move each examination to a new random feasible time-slot.
6. Five examinations at random and move each examination to a new random feasible time-slot.
7. One move of 1-pair Kempe-chain of one random examination.
8. One move of 1-pair Kempe-chain of one highest penalty examination selected from a random 10% selection of the examination.
9. One move of 1-pair Kempe-chain of one highest penalty examination selected from a random 20% selection of the examination.
10. Two moves of 1-pair Kempe-chain of one random examination.
11. Two moves of 1-pair Kempe-chain of one highest penalty examination selected from a random 10% selection of the examination.
12. Two moves of 1-pair Kempe-chain of one highest penalty examination selected from a random 20% selection of the examination.
13. Two time-slots at random and swap between them.

14. One time-slot at random and move to a new feasible time-slot.

15. Shuffle all time-slots at random.

The neighbourhood structures (1 to 15) were employed with the Toronto benchmark datasets, while the ITC2007 benchmark datasets employed the neighbourhood structures 1 to 9 only. This is because, in order to reduce the running time whenever a move was employed, a new best room need to be searched i.e. the best room that has the lowest penalty value of all rooms.

6.2.3 Acceptance Criteria of VNS

Algorithm 16 below illustrates the pseudo-code of the VNS algorithm with the great deluge algorithm as acceptance criteria. The great deluge algorithm is a local search approach that accepts worst solution based on an acceptance level of quality, B and simultaneously B is decreased with a certain amount called decay rate, β. This approach has demonstrated good performance within the timetabling problem. Implementation of these approaches is exemplified in studies in examination timetabling (McCollum et al., 2009, Özcan et al., 2010, Turabieh and Abdullah, 2011) and in course timetabling (Burke et al., 2003a, Landa-Silva and Obit, 2008). The pioneer of this approach, Dueck (1993), suggested that the decay rate, β should be low enough to allow the algorithm to search for more regions. Nevertheless, various implementations on the decay rate have been investigated by Turabieh and Abdullah (2011) (dynamic change of decay rate based on the 'electromagnetic-like mechanism'), by Landa-Silva and Obit (2008) (non-linear decay rate), and by McCollum et al. (2009) (re-heating decay rate).

In order to ease implementation, the decay rate, β for this study is set as 0.001 for the Toronto benchmark datasets and 0.05 for the ITC2007 benchmark datasets. This is based on the initial test and it is found that these values are the best setting depending on how much the acceptance level of quality, B should be reduced for each problem instance. On the other hand, the acceptance level of quality, B is initialised as the initial solution quality of $f(s)$ as discussed in Dueck (1993).

With reference to Algorithm 16, let s be an initial solution and is set as the best solution, s_{best}, obtained so far. The quality of solution s, $f(s)$ is set as $f(s_{best})$. While the algorithm starts the search, a solution s' is generated randomly by visiting the k^{th} neighbourhood sequentially until a local optima s'' is found. The solution s'' is accepted whenever the solution quality of s'', $f(s'')$ is better than $f(s_{best})$. Otherwise, if $f(s'')$ is better than the acceptance level of quality, B, then the solution s'' is accepted. The acceptance level of quality, B is updated by reducing it with a decay rate, β. Every time the solution quality, $f(s)$ is accepted, the search will continue with the identified neighbourhood, k as defined in Section 6.2.1.

Algorithm 16 Acceptance criteria of VNS algorithm

```
Select the set of neighbourhood structures $N_k, k = 1, ..., k_{max}$, to be used in the search;
choose stopping condition;
set initial solution $s$; $s_{best} \leftarrow s$; $f(s_{best} \leftarrow f(s))$;
Estimate the acceptance level of quality to be accepted, $B = f(s)$; set the decay rate,
$\beta$;
Repeat until stopping criteria is satisfied:
1. Set $k := 1$;
2. Until $k = k_{max}$, repeat:
(a) Shaking: Generate a point $s'$ at random from the $k^{th}$ neighbourhood of $s(s' \in N_k(s))$;
Repair mechanism:
(b) Local Search: Apply a local search method with $s'$ as initial solution until local
optimum $s''$ is obtained.
(c) Move or not:
Calculate $f(s'')$
Great deluge acceptance criteria:
if $f(s'')$ is better than $f(s_{best})$ then
  $s \leftarrow s''$
  $s_{best} \leftarrow s''$
else
  if $f(s'')$ is better than $B$ then
    $s \leftarrow s''$
  end if
end if
$B = B - \beta$
Continue the search with identified $k$.
```

The approach is incorporated with a repair mechanism since infeasible moves are considered. The repair mechanism for the Toronto benchmark datasets is shown in Algorithm 17. Each examination is considered to be moved to the time-slot that could be reduced to the lowest delta_cost. The examination with the lowest delta_cost to be moved, is moved to the best time-slot, j_{best}, and the process continues until no delta_cost is incurred for further improvement.

The repair mechanism for the ITC2007 benchmark datasets is illustrated in Algorithm 18. It does not, however, consider all examinations to be repaired: only a number of examinations are considered, including those in the list of G that are related to the infeasible moves. This is to avoid long running times during the repair process involved when searching for new best room each time the delta_cost is calculated. In any circumstance, if the delta_cost could reduce the current penalty cost, then the examination e is moved to time-slot j and their best room. The repair mechanism then is restarted

Algorithm 17 Repair mechanism for the Toronto benchmark datasets

```
Repair mechanism:
while the delta_cost $> 0$ do
  for $i = 1$ to number of examinations do
    for $j = 1$ to number of slots do
      Find the lowest delta_cost;
      $i_{best} \leftarrow i$;
      $j_{best} \leftarrow j$;
    end for
  end for
  Move $i_{best}$ to $j_{best}$
end while
```

with $i = 0$. The number of examinations to be repaired is only one percent of the overall examinations and if there is no improvement for a certain time then the number of examinations is increased to two percent.

Algorithm 18 Repair mechanism for the ITC2007 benchmark datasets

```
Repair mechanism:
$G$ ={list of examinations related with infeasible move}
Set number of examinations, $numEx$ to be repaired where the examination from 1
to $n$ are chosen randomly from all examinations and the remaining examinations i.e.
from $n + 1$ to $numEx$ are examinations related with the infeasible move
for $i = 1$ to $numEx$ do
  if $i < n$ then
    Choose $e$ randomly from all examinations
  else
    Choose $e$ from $G$
  end if
  for $j = 1$ to number of time-slots do
    Find the best room of examination $e$ and best slot $j$
    Calculate delta_cost
    if $delta\_cost < 0$ then
      Move $e$ to $j$ and the best room
      $i = 1$
    end if
  end for
end for
```

6.3 Experiments and Results

The stopping conditions for the experiments were set as 50000 iterations for the Toronto benchmark datasets, while the ITC2007 benchmark datasets followed the running time stated in the competition rules. However, the running time for the initial solutions is not included during the improvement phase due to the reason of comparing the performance of the algorithm using different initial solution values. A hundred runs were obtained for each dataset tested with three different initial solutions and with different types of neighbourhood orderings. The results are provided in the tables below, each table representing the results for each initial solution tested with different neighbourhood orderings. The best solution for each dataset is represented in bold font.

6.3.1 Toronto

Tables 6.1, 6.2 and 6.3 illustrate the results of poor, good and multiple initial solutions respectively, tested with different neighbourhood orderings for the Toronto benchmark datasets. The initial values of the poor and good solutions for each datasets are also provided in the tables. It is intended that these values will be improved using the VNS-GD algorithm, repeating each value for a hundred runs. On the other hand, since the multiple initial solution is generated at each run before proceeding to the improvement phase, only the average value of all generated initial solutions is provided in Table 6.3.

The overall results of poor initial solution illustrates in Table 6.1 show that the basic VNS that always starts with $k = 0$ whenever there is improvement to the solution quality, generates most of the best solutions with four best results and two ties for kfu93 and sta83 I. With other neighbourhood orderings, start-k obtained four best results with one tie, adaptive I obtained one best result and one tie with other types of neighbourhood orderings, adaptive II obtained one best result but tied with basic VNS and RL obtained one best result and a tie for sta83 I. The standard deviation (of less than one) for different neighbourhood orderings are relatively small.

The results generated using the good initial solution indicate the same pattern as that for the poor initial solution, where the basic VNS obtained best results of six problems and a tie for sta83 I. On the other hand, the other neighbourhood ordering types obtained three best results for start-k, one best result for the adaptive II and RL. The rest of neighbourhood orderings are tied for sta83 I. Considering the standard deviation value, the good initial solution provides less than the poor initial solution. The poor initial solution contributes higher standard deviation and this is due to the poor starting-point which tends to generate variation in the final solution quality.

The results from the multiple initial solutions also appear to work well with this approach, indicating the same pattern when the solutions are generated. The basic VNS shows great success with best solutions of six problems, while start-k, adaptive I and RL, each obtained one best results respectively while adaptive II obtained two best results. The rest of the neighbourhood orderings are tied for the sta83 I problem. The standard

Table 6.1: The results of poor initial solution tested with different neighbourhood orderings for the Toronto benchmark datasets (RL = reinforcement learning, stdev. = standard deviation, Av. t(m) = average running time in minutes)

		One poor initial solution						
Problem	Initial value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Av. t(m)	Best
car91	6.35	4.95	**4.89**	4.97	4.97	4.97	518.95	4.89
stdev.		0.06	0.06	0.06	0.07	0.07		
car92	5.43	4.19	4.18	**4.14**	4.15	4.18	274.76	4.14
stdev.		0.07	0.06	0.06	0.05	0.05		
ear83 I	47.85	**33.45**	33.55	33.52	33.63	33.60	18.56	33.45
stdev.		0.52	0.59	0.51	0.51	0.55		
hec92 I	13.91	10.29	**10.21**	10.25	10.38	10.33	1.33	10.21
stdev.		0.15	0.14	0.14	0.14	0.14		
kfu93	18.03	13.48	**13.46**	13.56	13.46	13.52	68.87	13.46
stdev.		0.15	0.15	0.15	0.15	0.14		
lse91	14.29	**10.53**	10.55	10.60	10.57	10.55	57.30	10.53
stdev.		0.19	0.19	0.15	0.17	0.19		
rye93	11.71	**8.37**	8.45	8.57	8.50	8.61	120.96	8.37
stdev.		0.10	0.10	0.10	0.11	0.11		
sta83 I	196.68	**157.06**	**157.06**	**157.06**	157.07	**157.06**	1.06	157.06
stdev.		0.06	0.05	0.06	0.06	0.05		
uta92 I	4.40	3.41	**3.40**	3.44	3.43	3.42	485.08	3.40
stdev.		0.05	0.04	0.03	0.04	0.04		
ute92	32.80	25.04	**24.96**	25.08	25.06	24.96	4.06	24.96
stdev.		0.10	0.10	0.11	0.11	0.10		
tre92	10.91	**8.18**	8.26	8.32	8.28	8.29	38.08	8.18
stdev.		0.11	0.13	0.09	0.10	0.10		
yor83 I	50.48	36.22	36.77	36.23	36.71	**36.05**	10.32	36.05
stdev.		0.40	0.40	0.39	0.36	0.44		

deviation among the neighbourhood orderings are clearly about the same, but there are some significant differences when compared with different types of initial solutions.

Among the problems, sta83 I shows almost no differences when employed with different types of neighbourhood ordering. Even though sta83 I starts with different initial solutions, the move is tended to get stuck at local optimum and no further improvement could be obtained. This may be due to the incorporation of the repair mechanism that checked and moved each examination and time-slot which could reduce the least penalty cost to the lowest level. Although there are variations in the standard deviation values for different initial solutions used for the sta83 I problem, the start with good initial solution shows that the standard deviations (less than 0.004) are very small.

The overall results demonstrate that the basic VNS performed very well where it obtained most best results of the thirteen problems of the Toronto benchmark datasets

Table 6.2: The results of good initial solution tested with different neighbourhood orderings for the Toronto benchmark datasets (RL = reinforcement learning, stdev. = standard deviation, Av. t(m) = average running time in minutes)

		One good initial solution						
Problem	Initial value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Av. t(m)	Best
car91	5.08	**4.86**	4.87	4.89	4.94	4.88	542.65	4.86
stdev.		0.03	0.03	0.03	0.02	0.03		
car92	4.34	4.24	4.22	4.22	4.21	**4.20**	286.64	4.20
stdev.		0.02	0.01	0.01	0.01	0.02		
ear83 I	36.91	34.36	33.81	34.08	**33.76**	34.17	17.62	33.76
stdev.		0.54	1.01	0.69	0.69	0.63		
hec92 I	11.13	**10.11**	10.19	10.28	10.23	10.30	1.38	10.11
stdev.		0.15	0.15	0.14	0.16	0.14		
kfu93	14.42	13.83	**13.62**	13.89	13.80	13.84	60.38	13.62
stdev.		0.06	0.09	0.05	0.07	0.06		
lse91	11.41	10.65	**10.58**	10.65	10.66	10.59	54.89	10.58
stdev.		0.12	0.13	0.14	0.12	0.11		
rye93	9.37	8.51	**8.45**	8.51	8.50	8.53	133.74	8.45
stdev.		0.08	0.08	0.08	0.09	0.08		
sta83 I	157.34	**157.08**	**157.08**	**157.08**	**157.08**	**157.08**	1.10	157.08
stdev.		0.00	0.00	0.00	0.00	0.00		
uta92 I	3.52	3.49	**3.46**	3.48	3.48	3.47	496.32	3.46
stdev.		0.00	0.01	0.01	0.01	0.01		
ute92	26.24	**24.95**	24.99	24.97	25.04	24.97	4.52	24.95
stdev.		0.12	0.13	0.11	0.11	0.13		
tre92	8.73	8.36	**8.28**	8.31	8.37	8.38	36.08	8.28
stdev.		0.06	0.08	0.06	0.06	0.06		
yor83 I	39.67	37.53	**36.51**	37.14	37.19	37.31	10.39	36.51
stdev.		0.35	0.52	0.42	0.38	0.39		

when tested with various initialisations. This suggests that the neighbourhood ordering with respect to the size of the neighbourhood can affect the search for good solutions. On each occasion that there is improvement to the solution quality, the search always starts with a small neighbourhood structure. This would allow the search to explore more regions that cannot be achieved by other larger neighbourhood structures, while at the same time it could reduce the processing time because the search always begins with a small size neighbourhood.

The running time for this approach is quite long because of the incorporation of a repair mechanism. The repair mechanism for the Toronto benchmark datasets considers each examination to be repaired or improved, taking into consideration a move to time-slot that can reduce the current penalty cost to the lowest level. In this study, the

Table 6.3: The results of multiple initial solutions tested with different neighbourhood orderings for the Toronto benchmark datasets (RL = reinforcement learning, stdev. = standard deviation, Av. t(m) = average running time in minutes)

	Av.	Multiple initial solution						
	initial		Basic	Adaptive	Adaptive		Av.	
Problem	value	Start-k	VNS	I	II	RL	t(m)	Best
car91	5.65	4.87	4.88	4.89	4.87	**4.83**	519.42	4.83
stdev.		0.08	0.07	0.08	0.07	0.08		
car92	4.93	4.10	**4.06**	4.07	4.13	4.12	299.9	4.06
stdev.		0.06	0.05	0.06	0.18	0.06		
ear83 I	41.94	33.43	**33.22**	33.38	33.53	33.41	19.49	33.22
stdev.		0.52	0.70	0.47	0.47	0.63		
hec92 I	12.76	10.25	10.27	10.28	**10.23**	10.35	1.35	10.23
stdev.		0.16	0.17	0.16	0.14	0.15		
kfu93	16.23	13.39	**13.30**	13.57	13.46	13.41	66.36	13.30
stdev.		0.18	0.17	0.13	0.15	0.16		
lse91	12.82	10.45	10.45	**10.36**	10.38	10.46	56.81	10.36
stdev.		0.25	0.27	0.26	0.25	0.28		
rye93	11.08	8.56	**8.42**	8.53	8.48	8.53	140.12	8.42
stdev.		0.09	0.11	0.13	0.10	0.12		
sta83 I	160.16	**157.04**	**157.04**	**157.04**	**157.04**	**157.04**	1.37	157.04
stdev.		0.08	0.08	0.07	0.07	0.07		
uta92 I	3.88	**3.36**	3.38	3.37	**3.36**	3.40	504.24	3.36
stdev.		0.04	0.05	0.05	0.05	0.05		
ute92	28.55	24.97	**24.92**	24.99	24.93	24.93	4.38	24.92
stdev.		0.12	0.13	0.09	0.09	0.09		
tre92	9.58	8.25	**8.12**	8.22	8.24	8.22	39.38	8.12
stdev.		0.11	0.13	0.11	0.11	0.14		
yor83 I	45.18	36.48	**35.88**	36.27	36.37	35.96	10.69	35.88
stdev.		0.35	0.50	0.42	0.40	0.38		

repair mechanism not only works for the infeasible moves but also tries to repair each examination assignment by reducing the assignment cost.

Tables 6.4 (a) and (b) illustrate the comparison of different improvement approaches in the literature with the VNS-GD approach. In order to see the best approach, the results of each problem are ranked and the best approach is identified based on the least average rank. The rank value of each approach is provided in brackets next to the solution quality in each table. From the average ranked, the VNS-GD approach is placed as the second best approach; however, it has not obtained a best result for any of the benchmark problems. The best approach with the least rank is represented by the study of Burke et al. (2010a) which employed the VNS approach with genetic algorithm. The results of pur93 I were not included in the previous tables since it required a long running time and it was almost impossible to obtain the results for a hundred runs due to the size of the problem. The run for purdue93 I was performed with only good initial solution starting with solution quality (5.74), and was repeated only three times. The best results of pur93 I is presented in Table 6.4.

6.3.2 ITC2007

The results of three different initialisations and different neighbourhood orderings of the ITC2007 benchmark datasets are presented in Tables 6.5, 6.6 and 6.7. Table 6.5 shows the performance of poor initial solution on different neighbourhood orderings, where it behaves effectively with neighbourhood ordering type adaptive I and II, for which each of these neighbourhood orderings obtained four best results. The RL neighbourhood ordering also performed well with two best results and one tie with adaptive II, while the basic VNS obtained only two best results and start-k do not yield any best result.

The performance of neighbourhood orderings are quite dissimilar when the good initial solution is used for improvement. Table 6.6 illustrates that start-k yields three best results while the basic VNS achieves only one best result. The adaptive neighbourhood ordering demonstrates a good performance, the adaptive II obtaining six best results while the adaptive I achieves one best result. The results of multiple initial solutions are presented in Table 6.7, which shows that adaptive I performed effectively when starting with multiple initial solutions with four best results. The other neighbourhood orderings also performed well where start-k and basic VNS obtained two best results each, while adaptive II and RL neighbourhood ordering each obtained one best result.

As can be observed in Table 6.7, the initial solutions for Exam_12 can not be obtained, nor do the repair mechanism work effectively to repair the infeasibility. It is worth noting that the conflict density of Exam_12 is the highest among the ITC2007 benchmark datasets with 18.45%. This may explain why the repair mechanism failed to place the unscheduled examination in the correct time-slot and room. Nevertheless, the repair mechanism works well on Exam_11. As Table 6.7 demonstrates, the average initial value of Exam _11 is slightly higher because some of the generated solutions were infeasible. The repair mechanism can fix the infeasibility for the unscheduled examinations, but to do this, a long running time is required. In the case of the ITC2007 benchmark datasets, the results that started with infeasible initial solutions were not encouraging due to the limitation of running time.

The standard deviation depicted in Tables 6.5, 6.6 and 6.7 reveals variations in the performance of the solution quality when implemented with different neighbourhood orderings. These variations may be caused by the characteristics of the benchmark

Table 6.4: Comparison of different improvement approaches with VNS-GD

Problem	[1]	[2]	[3]	[4]	[5]	[6]
car91	6.2 (10)	-	4.65 (3)	5.1 (6)	**4.5** (1)	5.4 (9)
car92	5.2 (10)	-	4.1 (4.5)	4.3 (7.5)	3.93 (2)	4.2 (6)
ears83 I	45.7 (12)	38.9 (11)	37.05 (10)	35.1 (8)	33.71 (4)	34.2 (5)
hec92 I	12.4 (12)	11.2 (10)	11.54 (11)	10.6 (6)	10.83 (8)	10.4 (5)
kfu93	18 (12)	16.5 (11)	13.9 (8)	13.5 (4.5)	13.82 (7)	14.3 (9)
lse91	15.5 (12)	13.2 (11)	10.82 (8)	10.5 (7)	10.35 (5)	11.3 (9.5)
pur93 I	-	-	-	-	-	-
rye92	-	-	-	8.4 (2)	8.53 (4)	8.8 (6)
sta83 I	160.8 (11)	158.1 (6)	168.73 (12)	157.3 (4.5)	158.35 (9)	157 (2)
tre92	10 (12)	9.3 (10)	8.35 (5)	8.4 (6.5)	7.92 (3)	8.6 (8.5)
uta92 I	4.2 (11)	-	3.2 (3.5)	3.5 (8)	**3.14** (1)	3.2 (3.5)
ute92	29 (12)	27.8 (11)	25.83 (7)	25.1 (4)	25.39 (6)	25.3 (5)
yor83 I	41 (12)	38.9 (10)	37.28 (8)	37.4 (9)	36.53 (7)	36.4 (6)
Av. Rank	11.15	10.38	7.62	6.35	5.04	6.38
Rank	12	11	9	6	5	7

(a)

Problem	[7]	[8]	[9]	[10]	[11]	VNS-GD
car91	5.2 (7)	5.2 (7)	6.6 (11)	4.6 (2)	4.8 (4)	4.83 (5)
car92	4.4 (9)	4.3 (7.5)	6 (11)	**3.9** (1)	4.1 (4.5)	4.06 (3)
ears83 I	34.9 (6)	36.8 (9)	**29.3** (1)	32.8 (2)	34.92 (7)	33.22 (3)
hec92 I	10.3 (4)	11.1 (9)	**9.2** (1)	10 (2)	10.73 (7)	10.11 (3)
kfu93	13.5 (4.5)	14.5 (10)	13.8 (6)	**13.0** (1.5)	**13.0** (1.5)	13.3 (3)
lse91	10.2 (4)	11.3 (9.5)	**9.6** (1)	10 (2)	10.01 (3)	10.36 (6)
pur93 I	-	4.6 (2)	**3.7** (1)	-	4.73 (3)	5.71 (4)
rye92	8.7 (5)	9.8 (8)	**6.8** (1)	-	9.65 (7)	8.37 (2)
sta83 I	159.2 (10)	157.3 (4.5)	158.2 (7)	**156.9** (1)	158.26 (8)	157.04 (3)
tre92	8.4 (6.5)	8.6 (8.5)	9.4 (11)	7.9 (2)	**7.88** (1)	8.12 (4)
uta92 I	3.6 (10)	3.5 (8)	3.5 (8)	3.2 (3.5)	3.2 (3.5)	3.36 (6)
ute92	26 (8)	26.4 (10)	**24.4** (1)	24.8 (2)	26.11 (9)	24.92 (3)
yor83 I	36.2 (3.5)	39.3 (11)	36.2 (3.5)	**34.9** (1)	36.22 (5)	35.88 (2)
Av. Rank	6.62	8.00	4.88	3.00	4.88	3.62
Rank	8	10	3.5	1	3.5	2

(b)

[1] Di Gaspero and Schaerf (2001), [2] Paquete and Fortseca (2001), ([3] Burke and Newall (2003), ([4] Merlot et al. (2003), [5] Yang and Petrovic (2004), [6] Côté et al. (2005), [7] Abdullah et al. (2007), [8] Eley (2007), [9] Caramia et al. (2008), [10] Burke et al. (2010a) and [11] Turabieh and Abdullah (2011)

datasets. However, the best results shown within each table indicate that the adaptive approach works effectively on the ITC2007 benchmark datasets.

TABLE 6.5: The results of poor initial solutions tested with different neighbourhood orderings for the ITC2007 benchmark datasets (RL = reinforcement learning, stdev. = standard deviation)

	Initial	One poor initial solution					
Problem	value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Best
Exam_1	13374	8404	8591	**8247**	8529	8403	8247
stdev.		202.05	208.30	239.25	184.71	220.39	
Exam_2	3557	582	563	**557**	577	562	557
stdev.		58.26	47.10	46.62	52.47	55.57	
Exam_3	23622	14440	**14330**	14444	14525	14629	14330
stdev.		826.40	1087.19	1432.19	863.75	1237.44	
Exam_4	26515	21429	21543	21148	**20929**	21810	20929
stdev.		368.42	359.75	366.64	391.21	383.92	
Exam_5	9608	4655	4733	4646	4639	**4589**	4589
stdev.		296.58	263.00	314.44	285.24	285.38	
Exam_6	34265	27395	27505	27405	**27270**	27350	27270
stdev.		613.93	960.80	693.82	767.69	700.90	
Exam_7	18726	5819	6015	5883	**5805**	5914	5805
stdev.		244.58	237.92	244.62	243.82	268.31	
Exam_8	23620	11152	**10947**	10982	11241	11246	10947
stdev.		1244.01	1258.25	1342.16	1163.68	1228.04	
Exam_9	2594	1303	1332	1262	1325	**1259**	1259
stdev.		46.67	44.66	60.83	45.63	56.64	
Exam_10	36725	15116	15210	**14343**	15276	15361	14343
stdev.		292.33	266.90	335.06	256.22	263.88	
Exam_11		43672	44404	**43345**	43614	43511	43345
stdev.	54674	1683.41	1335.58	2377.75	1702.52	1483.07	
Exam_12		6361	6361	6361	**6285**	**6285**	6285
stdev.	7273	100.21	105.74	107.06	85.73	101.27	

Comparison of the VNS-GD results with other approaches within the ITC2007 benchmark datasets shows that the VNS-GD does not yield any best results. However, placed as the fifth best approach, the results are competitive with other approaches.

6.3.3 Discussions

Overall, the performance of the ITC2007 benchmark datasets differs from the Toronto benchmark datasets in its type of initialisation: the ITC2007 benchmark datasets performed effectively with an adaptive approach (either adaptive I or II) while the Toronto benchmark datasets, in most cases, performed well with the basic VNS. The significant

Table 6.6: The results of good initial solutions tested with different neighbourhood orderings for the ITC2007 benchmark datasets (RL = reinforcement learning, stdev. = standard deviation)

	Initial	One good initial solution					
Problem	value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Best
Exam_1	11019	**6887**	7043	7085	6984	6913	6887
stdev.		193.84	191.18	179.59	169.88	213.29	
Exam_2	2880	**631**	**631**	642	**631**	625	631
stdev.		29.96	38.12	53.68	60.30	47.75	
Exam_3	19098	**11550**	11600	11919	11808	11659	11550
stdev.		485.67	486.29	388.16	984.76	1031.55	
Exam_4	20830	18034	18583	17949	**17891**	17960	17891
stdev.		222.98	134.42	239.14	308.82	210.38	
Exam_5	7975	**4554**	4672	4631	4729	4680	4554
stdev.		147.85	129.85	181.18	128.36	211.25	
Exam_6	28330	**26305**	26325	26380	26345	26290	26305
stdev.		90.47	74.46	66.73	71.28	89.51	
Exam_7	15573	6132	**6087**	6168	6204	6072	6087
stdev.		185.93	218.25	192.13	168.98	209.59	
Exam_8	19684	10678	10676	**10608**	**10608**	10634	10608
stdev.		581.42	593.23	589.76	605.52	623.62	
Exam_9	2157	1248	1264	1239	**1233**	**1233**	1233
stdev.		26.06	27.17	35.93	50.18	30.93	
Exam_10	16516	14989	15084	15042	**14906**	14926	14906
stdev.		162.92	114.58	121.40	126.56	141.47	
Exam_11	44921	36968	37113	36537	**36300**	36338	36300
stdev.		1204.67	927.76	1101.87	1193.64	1861.44	
Exam_12	5858	5624	5632	5610	**5610**	5624	5610
stdev.		34.84	32.79	38.56	43.02	35.21	

difference between the features and constraint requirements of these two datasets clearly suggests different treatment.

Figure 6.3 below illustrates the box-plot of different initial solutions of basic VNS applied to the Toronto benchmark datasets. In most problems, the good initial solution has lower variation compared with other types of initial solution, (with the exception of ear83 I) while the multiple initial solutions demonstrate the largest variation of all. In the case of the Toronto benchmark datasets, to start with different initial solutions offers the possibility to obtain good solution quality compared with using only a single initial solution, since diverse starting points could create different search features.

Figures 6.4 (a) and (b) illustrate the search movement of the first run for the basic VNS for the hec92 I and lse91 instances with different types of initial solutions. For hec92 I, the poor and multiple initial solutions show a sharp drop at the beginning of the

Table 6.7: The results of multiple initial solutions tested with different neighbourhood orderings for the ITC2007 benchmark datasets (RL = reinforcement learning, stdev. = standard deviation)

	Av. Initial	Multiple initial solution					
Problem	value	Start-k	Basic VNS	Adaptive I	Adaptive II	RL	Best
Exam_1	12473.86	**8026**	8329	8089	8067	8120	8026
stdev.		327.87	294.00	270.20	282.47	379.56	
Exam_2	4052.18	663	**633**	661	657	636	633
stdev.		81.36	78.03	73.78	78.36	86.66	
Exam_3	25340.86	13901	13842	**13657**	13725	14060	13657
stdev.		728.87	846.38	994.32	529.48	798.05	
Exam_4	35276.19	**20778**	22213	23508	20858	22705	20778
stdev.		3373.91	4301.51	3912.41	5160.88	2845.69	
Exam_5	9153.17	4527	4448	4467	**4420**	4626	4420
stdev.		214.91	207.60	233.33	249.18	202.21	
Exam_6	29150.06	26685	26410	26480	26525	**26385**	26385
stdev.		207.79	266.07	265.54	240.75	263.97	
Exam_7	17068.30	6147	**5972**	6052	6072	6213	5972
stdev.		245.31	247.73	193.80	200.73	190.28	
Exam_8	22200.98	10989	10903	**10310**	10618	10788	10310
stdev.		291.57	348.00	400.97	297.41	323.67	
Exam_9	2283.62	1269	1266	1260	**1251**	**1251**	1251
stdev.		72.50	65.12	48.48	54.03	55.84	
Exam_10	17852.39	15095	14927	**14826**	15286	15168	14826
stdev.		316.17	328.78	260.60	265.95	275.36	
Exam_11	106586.57	49701	46468	**42669**	47363	46150	42669
stdev.		47998.54	12896.80	11001.79	11692.19	11608.70	
Exam_12	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*	*inf.*
stdev.		-	-	-	-	-	

search. This is due to the employment of a mechanism that repaired each examination by reducing their assignment cost. As can be observed, the poor initial solution is gradually moved to improve the solution quality throughout the iteration and started to get stuck half way through out the search until the search is finished. On the other hand, the multiple initial solution shows that the search stuck at the beginning of the iterations, taking a longer searching time to jump to the other region; the solution quality is then successfully reduced until the iterations end. The good initial solution already started with a good solution quality at the beginning of the search, and the solution quality is improved by reducing the value gradually. For the Toronto benchmark instances, the search for the good initial solution tends to get stuck easily, and for some problems, only little improvement can be obtained until the end of the iterations. Since the solution is already started with a good initial solution, the cannot jump out of its local optima. This is due to the lack of ability of the neighbourhood structure that was used in the

Table 6.8: Comparison of different approaches for ITC2007 benchmark datasets (The bold entries indicate the best results).

Problem	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	VNS-GD
Exam_1	4370	5905	8006	6670	12035	4370	4633	8559	6235	4775	**4368**	6887
	(2)	(6)	(10)	(8)	(12)	(2)	(4)	(11)	(7)	(5)	(1)	(9)
Exam_2	400	1008	3470	623	3074	**385**	405	830	2974	385	390	557
	(4)	(9)	(12)	(7)	(11)	(1)	(5)	(8)	(10)	(1)	(3)	(6)
Exam_3	10049	13862	18622	-	15917	9378	9064	11576	15832	**8996**	9830	11550
	(5)	(8)	(11)	(12)	(10)	(3)	(2)	(7)	(9)	(1)	(4)	(6)
Exam_4	18141	18674	22559	-	23582	**15368**	15663	21901	35106	16204	28924	17891
	(5)	(6)	(8)	(12)	(9)	(1)	(2)	(7)	(11)	(3)	(10)	(4)
Exam_5	2988	4139	4714	3847	6860	2988	3042	3969	4873	**2929**	3022	4420
	(2)	(8)	(10)	(6)	(12)	(2)	(5)	(7)	(11)	(1)	(4)	(9)
Exam_6	26950	27640	29155	27815	32250	26365	25880	28340	31756	**25740**	25995	26305
	(6)	(7)	(10)	(8)	(12)	(5)	(2)	(9)	(11)	(1)	(3)	(4)
Exam_7	4213	6683	10473	5420	17666	4138	**4037**	8167	11562	4087	4067	5805
	(5)	(8)	(10)	(6)	(12)	(4)	(1)	(9)	(11)	(3)	(2)	(7)
Exam_8	7861	10521	14317	-	16184	7516	**7461**	12658	20994	7777	7519	10310
	(5)	(7)	(9)	(12)	(10)	(2)	(1)	(8)	(11)	(4)	(3)	(6)
Exam_9	1047	1159	1737	1288	2055	1014	1071	-	-	**964**	-	1233
	(3)	(5)	(8)	(7)	(9)	(2)	(4)	(11)	(11)	(1)	(11)	(6)
Exam_10	16682	-	15085	14778	17724	14555	14374	-	-	**13203**	-	14343
	(7)	(10.5)	(6)	(5)	(8)	(4)	(3)	(10.5)	(10.5)	(1)	(10.5)	(2)
Exam_11	34129	43888	-	-	40535	31425	29180	-	-	**28704**	-	36300
	(4)	(7)	(10)	(10)	(6)	(3)	(2)	(10)	(10)	(1)	(10)	(5)
Exam_12	5535	-	5264	-	6310	5357	5693	-	-	**5197**	-	5610
	(4)	(10)	(2)	(10)	(7)	(3)	(6)	(10)	(10)	(1)	(10)	(5)
Av. Rank	4.33	7.63	8.83	8.58	9.83	2.67	3.08	8.96	10.21	1.92	5.96	5.75
Rank	(4)	(7)	(9)	(8)	(11)	(2)	(3)	(10)	(12)	(1)	(6)	(5)

[1] Müller (2008), [2] Gogos et al. (2008), [3] Atsuta et al. (2008), [4] De Smet (2008), [5] Pillay (2008), [6] Müller (2009), [7] McCollum et al. (2009), [8] Pillay (2010a), [9] Burke et al. (2010f), [10] Gogos et al. (2010a), [11] Turabieh and Abdullah (2011)

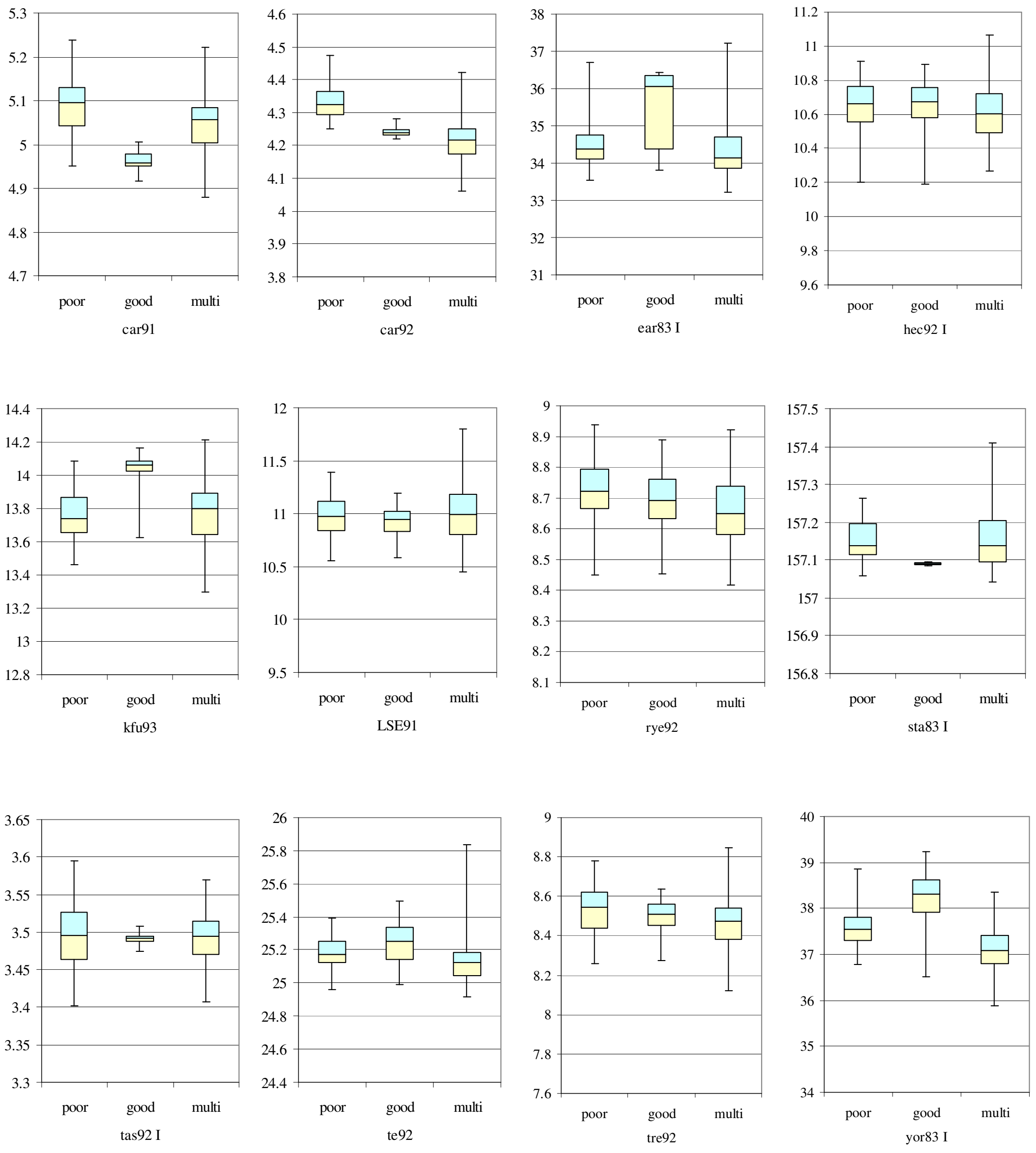
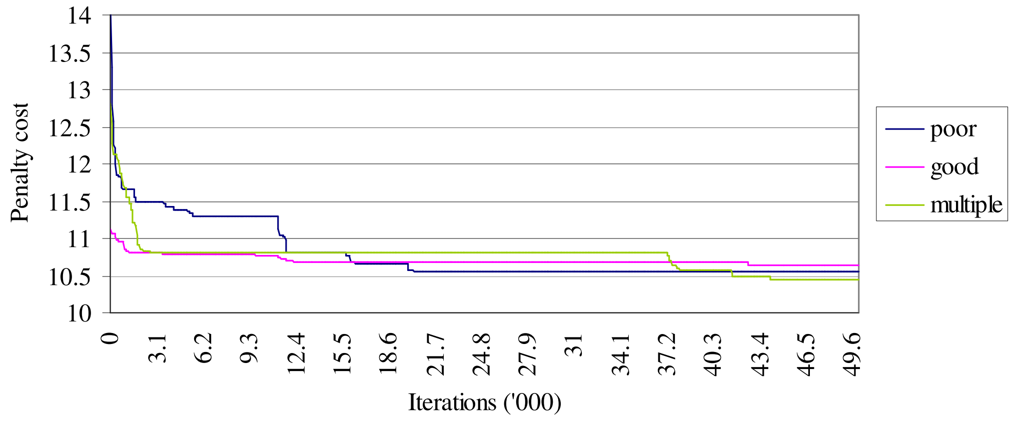


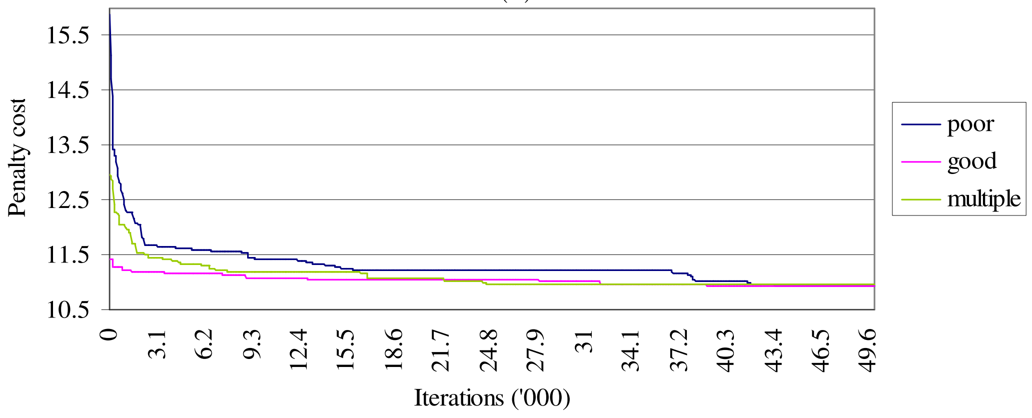
FIGURE 6.3: Box-plot of different initial solutions for the Toronto benchmark datasets

study. In that case, a drastic move should be performed in order to jump from its local optima in order to obtain better results. However, the jump should be carefully developed so that it is not too far from the current best solution and at the same time, it could reduce the running time while searching for good solutions.

The lse91 instance demonstrates almost the same behaviour of movement as the hec92 I instance. However, the good initial solution of the lse91 problem could obtain the best solution quality compared with other types of initial solutions. Although the search tends to get stuck, the solution quality could still be reduced from time to time until the end of iterations. The multiple initial solution is also competitive with the good initial solution. However, it has a tendency to get stuck in the middle of the search. Meanwhile, the poor initial solution continues to improve throughout the iterations and is stuck in the middle of the search. However, the stuck point is avoided after a number of iterations and the solution quality continues to improve.



(a)



(b)

FIGURE 6.4: The search movement of different initialisation for the (a) hec92 I and (b) lse91 of Toronto benchmark datasets

Figure 6.5 illustrates the box-plot of different initial solutions for the ITC2007 benchmark datasets. Most of the results for each dataset appear to have small variations when implemented with different initialisations, except for Exam_6, Exam_7 and Exam_10. It can be observed that, when starting the improvement phase with a good initial solution, there is a tendency to obtain a good solution at the end of the iterations. This is shown by almost all datasets except for Exam_2, Exam_5 and Exam_7. Starting with multiple initial solutions also demonstrates good performance when some of the datasets, for example, Exam_5, Exam_7 and Exam_10, obtained best results with this type of initialisation. Since the multiple initial solution failed to generate a feasible solution for Exam_12 when implemented with the improvement approach, the box-plot of Exam_12 shows only two types of initialisation, with the exception of multiple initial solutions.

The search movement of three different initialisations during the first run for two different datasets of the ITC2007 benchmark is illustrated in Figures 6.6 (a) and (b). A sharp drop in the solution quality during the first few iterations can be seen in both datasets

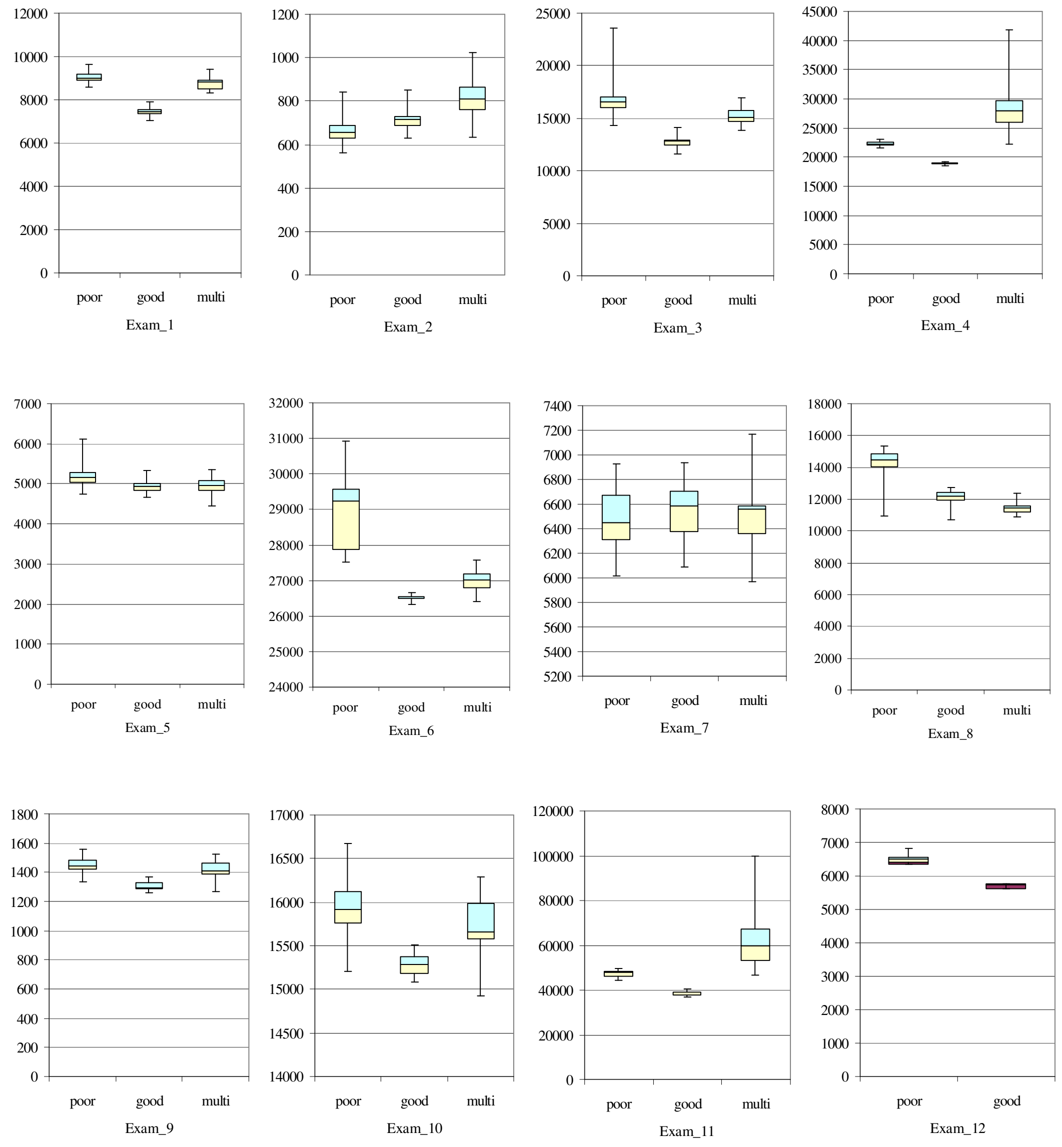


FIGURE 6.5: Box-plot of different initial solutions for the ITC2007 benchmark datasets

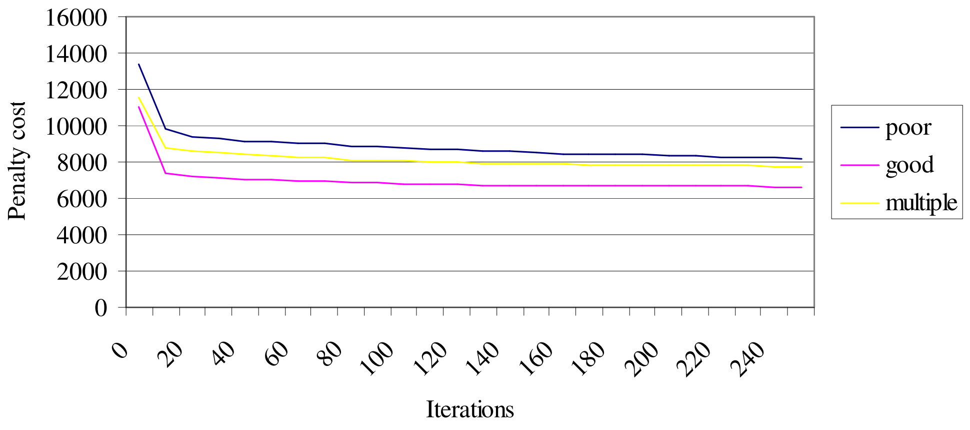
and it can be concluded that this behaviour was caused by the incorporation of a repair mechanism. This is also shown for the Toronto benchmark datasets. However, after a while, the improvement to the solution quality is too small, and it easily stuck during the search.

Figure 6.6 (a) indicates that starting with a good initial solution has an advantage in that the penalty cost continues to decrease from time to time. In the case of Exam_1, starting with a good initial solution could reduce the running time in order to achieve optimal solution. On the other hand, while starting with poor or multiple initial solution did lead to some improvement to the solution quality, it would take longer to achieve optimal solutions. It should be noted that as Exam_1 is a large dataset with 607 examinations and 7891 student enrolments, it may take a while for the algorithm to search for solutions and for the incorporation of a repair mechanism. In this case, it will take longer for the solution to achieve an optimal solution and there will be an obvious advantage in starting with a good initial solution for this size of problem. Figure 6.6 (b) illustrates different behaviour of search movement for Exam_9. Although it starts with a good initial solution, the search tends to get stuck during the search and could not improve further, while the multiple initial solution appeared to work effectively, improving the solution quality better than when using a good initial solution. However, the behaviour of the poor initial solution for Exam_9 is the same as that for Exam_1, requiring a longer time for the approach to achieve an optimal solution. It can be noted that Exam_9 is a smaller dataset with only 169 examinations and 655 student enrolments. This characteristic of Exam_9 may explain why the good initial solution performs differently from that of Exam_1.

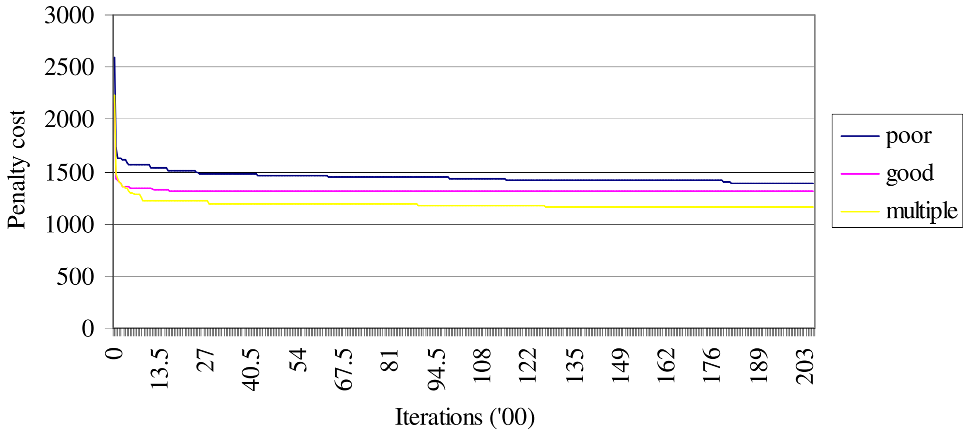
The incorporation of a repair mechanism enables the algorithm to spend more time in repairing and improving examination assignment. As shown in Figure 6.6, the results of Exam_1 are obtained with less than a thousand iterations, while Exam_9 is performed with more than twenty thousand iterations. The size of a dataset clearly affects the search for solutions. Since the runs of the ITC2007 benchmark datasets are required to follow the running time stated in the competition rules, only a small number of iterations could therefore be obtained by the larger datasets. It is believes that when longer running time is allocated, the results of the ITC2007 benchmark datasets could be further improved.

Observation on the results of different benchmark problems shows that different types of initialisation can shows different performance during the improvement phase in the multistage setting particularly for the VNS approach in this study. The results of the Toronto benchmark datasets are found to improve well with the multiple generated initial solution while for the ITC2007 benchmark datasets, it is good to start with good initial solutions. The results from different datasets give different performance on different types of initialisation. The Toronto benchmark datasets does not performed well with good initial solution. This is due to commencing with really good initial solutions which make the search to get stuck during the process even though a mechanism is employed to solve the problem. The multiple initial solutions with variety of solution qualities can assist in finding better solution. Meanwhile, the search for the poor initial solution is started very far from the good solution and will take a longer time for solution search. On the other hand, it is observed that the solution quality of ITC2007 can be much improved and faster when starting with good initial solutions, but this is different from the Toronto scenario. The ITC2007 benchmark datasets have different features in terms of the number of hard and soft constraints and these datasets are much more complicated when compared with the Toronto benchmark datasets. Meanwhile, the poor and the multiple generated initial solutions for the ITC2007 do not performed as well as good initial solutions within the given processing time.

Previous studies have shown that a good initial solution can help to produce better final solutions (Burke and Newall, 2003, Gogos et al., 2010a). Motivated by the studies, this research is focused on searching for good quality solutions during the timetable construction and, further, the initial solutions are improved in the improvement phase. In this study, the initial solutions are categorised into three major groups which are good, poor and multiple initial solutions. Nevertheless, the results show that starting the improvement phase with a good initial solution is not necessarily result in a good final solution. The study found that the performance of two datasets can differ and this may be due to the characteristic of the datasets. It can be concluded here that the utilisation of complex datasets can yield in the best performance with a good initial solution.



(a)



(b)

Figure 6.6: The search movement of different initialisation for (a) Exam_1 and (b) Exam_9 of ITC2007 benchmark datasets

6.4 Conclusion

This chapter aimed to improve the previously constructed solutions and at the same time to investigate the influence of various initialisations and neighbourhood orderings to the solution quality. The variable neighbourhood search - great deluge algorithm was presented and the experimental results showed that the approach could effectively improve various initialisations. In order to diversify the search, the study accepted infeasible moves and this infeasibility was repaired by a mechanism that was incorporated in the algorithm. At the same time, the repair mechanism worked simultaneously to further improve the solution quality, although the computational time was very expensive. Various initialisations and neighbourhood ordering influenced the solution quality, the effect of this depending on various characteristics of the implemented problems. The multiple initial solution demonstrates good performance to the solution quality of the Toronto benchmark datasets since starting the solution at different initial points can vary the search process and at the same time offers the possibility of obtaining good solution quality. On the other hand, the ITC2007 performed the best with a good initial solution. Note that, the ITC2007 is a complex problem with various types of constraints. In considering the neighbourhood orderings, the Toronto benchmark datasets performed the best with a strategy that always started with a small neighbourhood structure whenever improvement occurred, while the ITC2007 benchmark datasets performed the best with adaptive approach i.e. either Adapive I or Adaptive II. The characteristics of the ITC2007 problem that were required to satisfy many hard and soft constraints simultaneously explain why it performed differently from the Toronto problem instances. The VNS-GD approach can effectively improve the solution quality on two different examination timetabling problems and the obtained results are comparable with other improvement approaches mentioned in the literature.

Chapter 7

Conclusions and Future Work

The research work in this thesis investigates approaches for solving two different examination timetabling problems, where the focus is on the initialisation and improvement strategies proposed in the early chapters. Section 7.1 presents overall conclusions for the work that has been carried out and Section 7.2 highlights some research directions for future work.

7.1 Research Summary

This thesis explored some key issues related to initialisation strategies based on squeaky wheel optimisation, graph colouring heuristics and decomposition. The research then undertook a further investigation into the use of an improvement strategy based on a variable neighbourhood search - great deluge approach. Two main components were explored relating to initialisation and neighbourhood ordering in order to identify their effectiveness in improving the initial solution quality. The approaches introduced in this thesis were tested over two different benchmark datasets: Toronto and ITC2007 benchmarks and the problem instances in each benchmark have their own distinct features capturing real world complexities. Generally, the ITC2007 benchmark datasets are considered to be more difficult since they have many simultaneous hard and soft constraints.

The investigation of the squeaky wheel optimisation combined with graph colouring heuristics (namely, saturation degree, largest degree, largest weighted degree and largest enrollment) is presented in Chapter 3. The incorporation of a shuffling strategy has shown an improvement in the overall performance of the approach. Two different strategies were studied based on block and top-window, where examinations were shuffled randomly within a group having close difficulty values. Various sizes of block and top-window were investigated and statistical analysis showed that a shuffling strategy could yield a better ordering of examinations to be scheduled. It was observed that different sizes of block or top-window provided a varied performance for the algorithm. It is therefore suggested that the size should not be too large since, clearly, shuffling the examinations in a larger 'chunk' could, obviously, almost randomise the examination ordering, thus discarding what has been learned during the previous stages. Moreover, the trade-off between the computation time and solution quality was also observed during the experiments. The approach produced better quality solutions, particularly over the Toronto benchmark problem instances when given a longer computational time. This was possibly because a heuristic modifier significantly increases the timetabling difficulty value for the examinations, ultimately causing a substantial change in their orderings. The study also demonstrated that different graph colouring heuristics with various heuristic modifiers can lead to a different performance and cause a different number of unscheduled examinations. The study showed that the saturation degree generated the best performance when compared to the other graph colouring heuristics regardless of the heuristic modifier used in the approach. This success of saturation degree is possibly due to its dynamic nature. Additionally, it is observed that the exponential heuristic modifier may improve the performance of the approach when compared to the custom, additive or multiplicative heuristic modifiers. Seeing that different heuristic modifiers yield different performances motivated the idea of combining graph colouring heuristics for measuring the difficulty of scheduling an examination during the construction of timetables. The graph colouring heuristics were alternated during the search for a good timetable and this process was found to be useful to improve the performance of the overall approach.

The squeaky wheel optimisation approach was further investigated in Chapter 4 where graph colouring heuristics (namely, saturation degree and largest degree) were combined with a heuristic modifier in a linear formula where the resulting weighted sum was used to generate an overall score/value to order the examinations. The experimental results revealed that the combination of multiple graph colouring heuristics with a heuristic modifier can outperform the single graph colouring heuristic in squeaky wheel optimisation. Three strategies to decide on the values of weights are tested using this framework. Firstly, weights were fixed for each component/parameter. It was observed that different combinations of weight values generated different performances. Using a higher weight value for the heuristic modifier against graph colouring heuristics enhanced the performance of the overall approach, in particular for the Toronto benchmark instances. On the other hand, the performance of the approach for the ITC2007 benchmark datasets was not consistent. Instead of fixing the weight values of each parameter, we investigated changing them automatically using three strategies i.e. dynamic weighting, linear weighting and reinforcement learning. The experimental results revealed that the dynamic strategy based on calculating a penalty cost for each assignment is the best strategy in changing the weight values for the parameters. Nevertheless, this strategy increases the computation time because of the penalty cost calculation at each time when an assignment is made.

The effective use of the information regarding the unscheduled examinations within a constructive heuristic framework processing partial solutions was further investigated in Chapter 5. An approach is proposed where the unscheduled and scheduled examinations are separated automatically into two subsets. The unscheduled examinations were considered to be difficult to schedule and they are attempted to be scheduled first at each iteration. Each subset was augmented with a suitable ordering that is based on graph colouring heuristics. A boundary set was introduced where the examinations in the set were merged or swapped with the difficult set. It was observed that the use of the boundary set improved the overall performance of the approach. The roulette wheel selection is embedded into the approach giving a higher chance to an examination with a higher score of difficulty value to be selected for timetabling. Statistical analysis showed that the boundary size and roulette wheel selection size have significant effects in improving an initially generated solution. The experiments showed that choosing the sizes for both parameters beyond ten does not greatly improve the performance. Overall, the constructive approaches undertaken in this thesis demonstrated that the dynamic graph colouring heuristic yielded better orderings when compared to the static type of graph colouring heuristics. Furthermore, it was observed that the dynamic graph colouring heuristic produces fewer unscheduled examinations throughout the timetabling process.

The solutions from the constructive approaches pointed out earlier are later used as initial solutions fed into an improvement approach; variable neighbourhood search - great deluge. More on the improvement approach is provided in Chapter 6. There are successful multistage algorithms reported in the literature combining constructive and improvement stages (Müller, 2009). The goal of this part of the study was to see the effects of combining different constructive algorithms for initialisation and improvement approach. The variable neighbourhood search - great deluge algorithm was chosen as the improvement approach to be investigated based on its successful performance in examination timetabling problems reported in the literature (Müller, 2009). The variable neighbourhood search approach is capable of escaping from local optima since the approach incorporates various neighbourhood structures. Moreover, as a threshold acceptance strategy the great deluge algorithm considers poor solutions for acceptance based on a level at any given time when a worsening solution is obtained during the search process. A strategy to accept worsening solution might produce an infeasible solutions. In order to overcome a resulting infeasibility in a solution, a repair mechanism was used to ensure feasibility with respect to the problem constraints and, at the same time, to improve the quality of the solution in hand. The experimental results showed that the repair mechanism was effective and resolved the infeasibilities for all problem instances except for Exam_12, an ITC2007 benchmark problem instance. This problem has been identified as the most difficult instance within the ITC2007 benchmark with regard to the conflict density values. Although the repair mechanism is effective, it is computationally expensive because of the need for checking each potential examination in order to find the best assignment that could reduce the penalty cost.

Three sets of initial solutions i.e. poor, good and multiple initial solutions generated by the constructive approaches introduced in the previous chapters were considered in order to observe their influence on the performance of the overall approach. An investigation on the effect of initialisation indicated that a good initial solution did not necessarily yield the best solution at the end and there is a possibility that the improvement approach could get stuck at local optima. A special mechanism was needed in order to escape. It was observed that the improvement approach could not much improve an initially constructed solution for some problem instances. Sometimes starting from a good solution saved time for the improvement approach in the search for a better solution (assuming the algorithm is stopped after converging to a value). On the other hand, the poor initial solution has the potential to be further improved by allowing exploration of the search space around it. However, the search would take longer since the search started far away from a promising solution. The reason for the good performance of the multiple initial solutions on the Toronto benchmark is that these solutions allow for better exploration of solution search when starting with different starting points. Different initialisations or starting points for solution search creates an advantage for the algorithm to work at different starting points, thus, allowing a greater possibility of finding good solutions. On the other hand, the approach performed differently for the ITC2007 benchmark. It has been found that the problem could achieve best solutions when the search starts from good initial solutions. This was possibly due to the large number of hard and soft constraints that have to be satisfied simultaneously and also the size of the search space.

An investigation on the effect of the neighbourhood orderings for the variable neighbourhood search on its performance was also performed. The neighbourhood structures were ordered according to their sizes relating to the number of reassigned examinations. Five neighbourhood ordering strategies were introduced. The study reveals that starting the search process using a small neighbourhood structure and then increasing the step size offers the advantage of exploring immediate ‘close’ neighbourhoods first and then diversifying using larger step sizes. The repair mechanism in the study acts as a hill climber that either improves the solution further or returns the same quality solution. This phenomenon was observed for the Toronto benchmark problem instances. On the other hand, the performance of the approach for the ITC2007 benchmark varies for different neighbourhood orderings. There was no clear winner for the choice of a neighbourhood ordering strategy.

The overall approach combining different initialisation strategies with an improvement approach generated a successful performance when compared to the other improvement approaches in the literature over the Toronto and ITC2007 benchmarks. However, the performance of the overall approach is sensitive to parameter settings, such as, the weighting strategy or setting the decay rate of the great deluge algorithm. More adaptation mechanisms and automated parameter tuning strategies should be investigated.

7.2 Future Work

Several major research directions can be further explored in order to improve the performance of the proposed approaches. A framework based on the squeaky wheel optimisation combined with the use of graph colouring heuristics is very promising especially when several graph colouring heuristics are alternated (as shown in Chapter 3) or combined in a linear way (as shown in Chapter 4) with a heuristic modifier during the timetable construction. It would be interesting to investigate other heuristic selection methodologies within hyper-heuristic methods to select from constructive heuristics and whether the graph colouring heuristics could be combined under a generational (on-line or off-line) hyper-heuristic framework instead of a squeaky wheel optimisation framework in order to construct examination timetables.

In addition, there could be further investigation into the importance of the difficulty value as an aspect in changing the examination ordering. Since the study undertaken in this thesis has increased only a static amount of difficulty, it would be worthwhile to investigate the amount of difficulty value that should be increased in order to modify the examination ordering, while at the same time demonstrating that the examination is more difficult. The difficulty value of an examination can be explored in order to ascertain whether the value can modify the overall examination ordering, or to observe if the examination will still be in the same position. In the latter case, the amount of increase in the difficulty value can be adjusted to ensure the position of the relevant examination changes in the overall examination ordering.

Using a weighted average of parameters in order to obtain new difficulty values in the proposed linear approach offers a better ordering before the assignment of timetable slots to the examinations. The dynamic change of weight values within a linear approach can be further improved. It would be an advantage if the weight values could be changed or modified automatically. A mechanism may be required to decide when to change and which weight to change in order that it can be used during the timetabling process. Furthermore, more graph colouring heuristics combined with a heuristic modifier could be considered during timetable construction as the proposed approaches employ a subset of existing graph colouring heuristics.

As described in Chapter 5, the approach grouped the examinations into two subsets based on the difficulty of scheduling the examinations. The size of the difficult set containing the examinations which are difficult to timetable increases until some point during the search process where the size remains steady. It might be worth investigating ways of deciding which examination should be released from the difficult set, indicating that these examinations are no longer difficult and can be placed into the 'easy' set.

The variable neighbourhood search - great deluge approach requires significant running time due to the repair mechanism used within the approach for ensuring each move complies with the constraints and improves the current solution quality. In the case of the Toronto benchmark datasets, we showed that starting the search from a good initial solution fails to achieve good results as compared to initialising using multiple initial solutions. Using a good initial solution potentially seems to be a better strategy for obtaining improved results with a low standard deviation when compared to the other types of initial solutions.

Previous analysis on the search trajectory of the variable neighbourhood search - great deluge approach showed that it tends to get stuck during the search process. Since, currently, the move acceptance uses a static decay rate, methods such as 'rerising' for changing the decay rate dynamically, depending on the search progress can be investigated further. Previous studies in the literature show that changing the decay rate dynamically may guide the search to the promising regions of the search space, disallowing premature convergence to a local optimum.

Furthermore, the investigation of the neighbourhood ordering suggests that starting with a small neighbourhood structure could assist in searching for an unexplored region of solution space. An investigation into the size of the implemented neighbourhood structure can be undertaken by deciding the point up to which the small or large neighbourhood should be used. As this study demonstrates, using the small neighbourhood could reduce the computational time, while the large neighbourhood structure requires longer computational time for moving the related examinations or time-slots. It is proposed that the small neighbourhood can be used, especially when the search is stuck, because this small neighbourhood structure could reach to an unexplored region of the solution space.

Further exploration of variable neighbourhood search within a hyper-heuristic framework holds some promise. The mixing of neighbourhood structures can be enhanced and a variable neighbourhood search may be a high level heuristic that chooses from a subset of neighbourhood structures automatically.

Appendix A

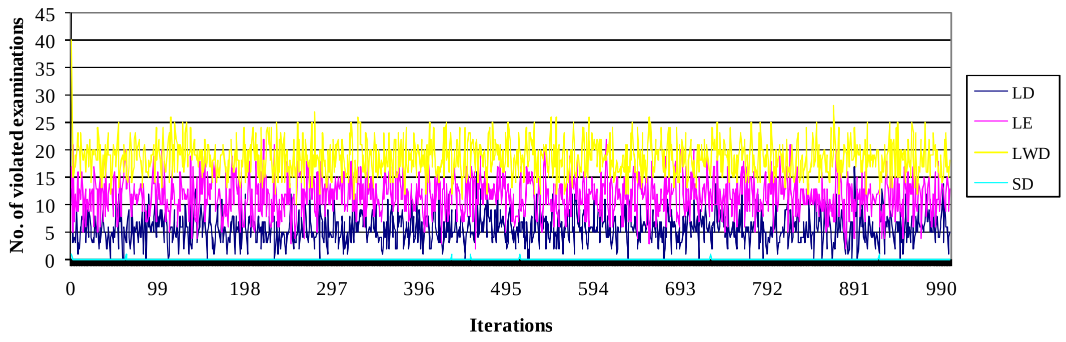
Graphs of Adaptive Heuristic Ordering Approach

A.1 Toronto Benchmark Datasets

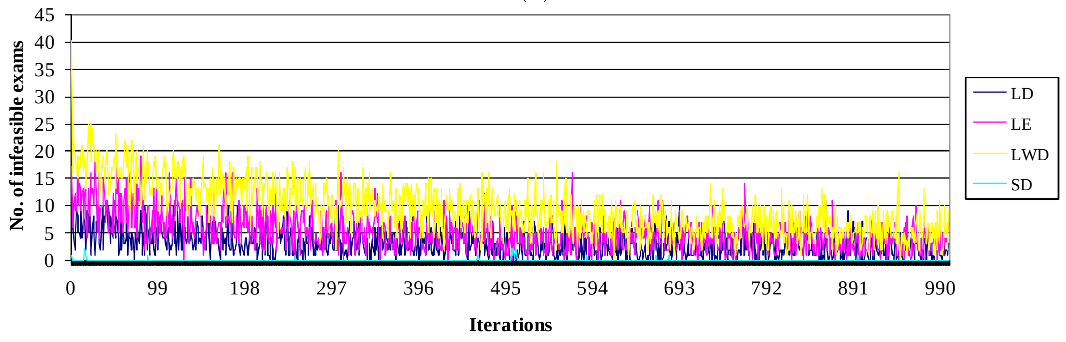
Figures A.1 to A.11 show the number of violated examinations at each iteration of the Toronto benchmark datasets with different heuristic modifier tested with basic AHO with top-window size five.

A.2 ITC2007 Benchmark Datasets

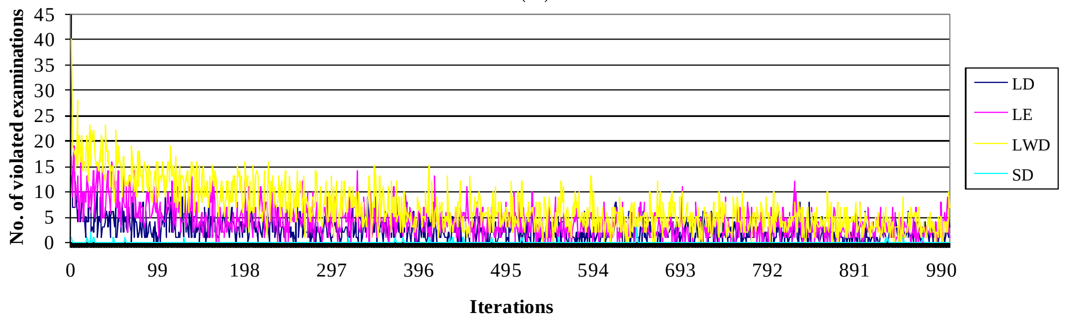
Figures A.12 to A.18 show the number of violated examinations at each iteration of the ITC2007 benchmark datasets with different heuristic modifier tested with basic AHO with top-window size five. Since the solutions of Exam_7 to Exam_10 are feasible during the first run, no graphs are provided for those instances.



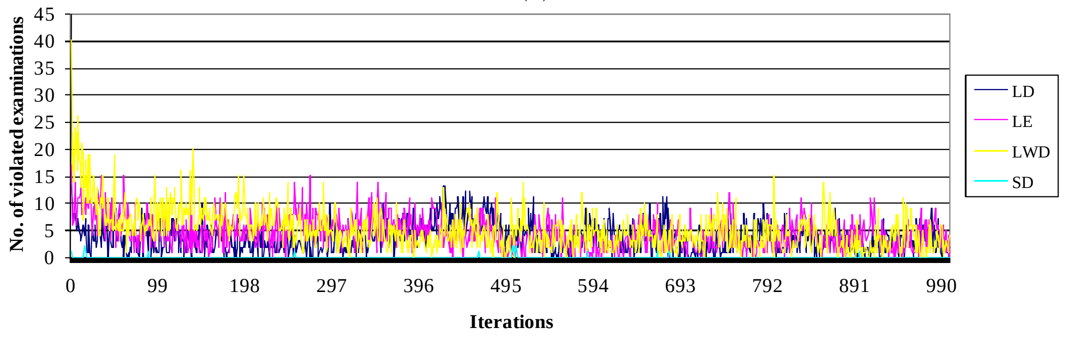
(a)



(b)

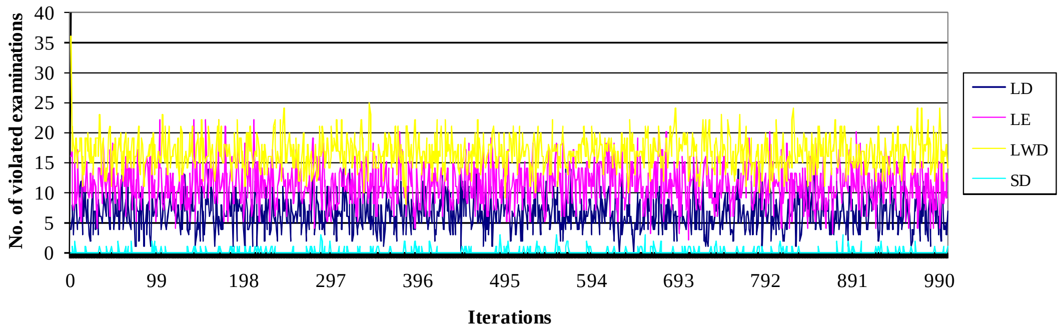


(c)

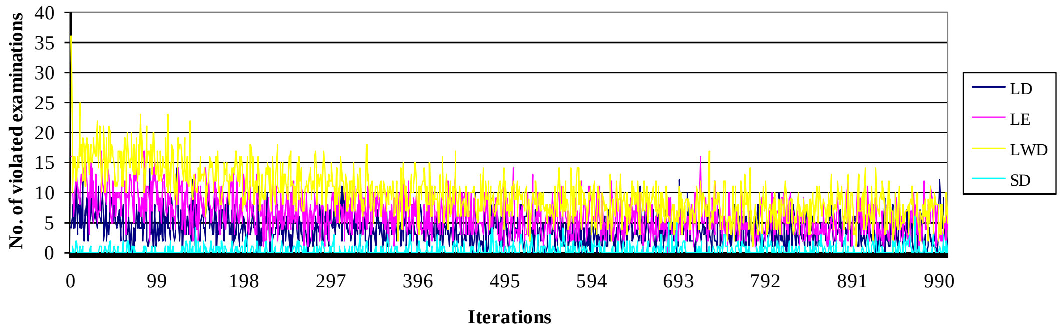


(d)

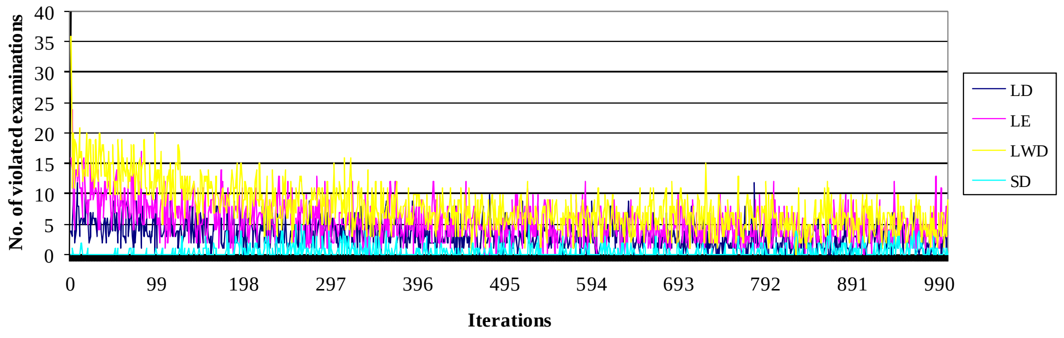
FIGURE A.1: The number of violated examinations at each iteration for car91 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



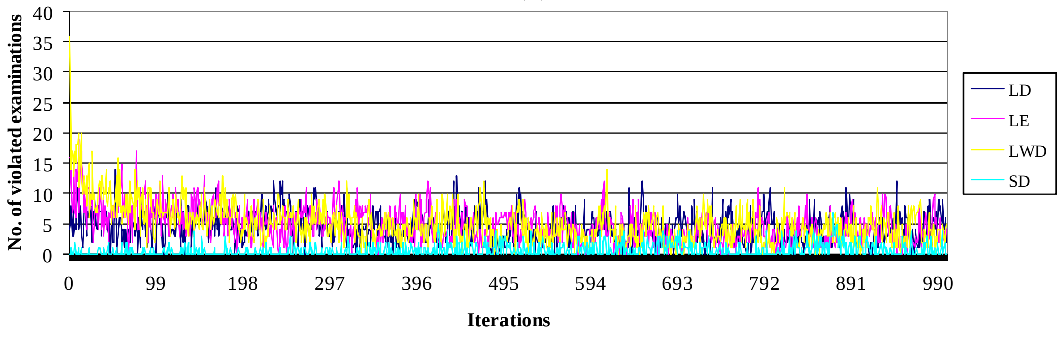
(a)



(b)

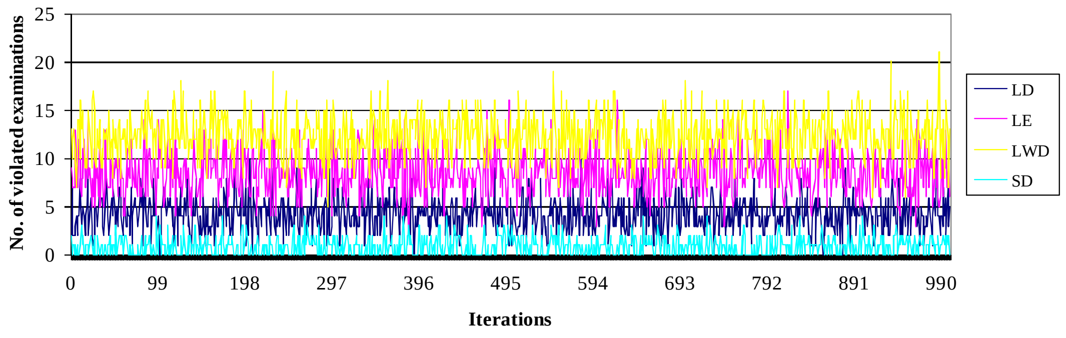


(c)

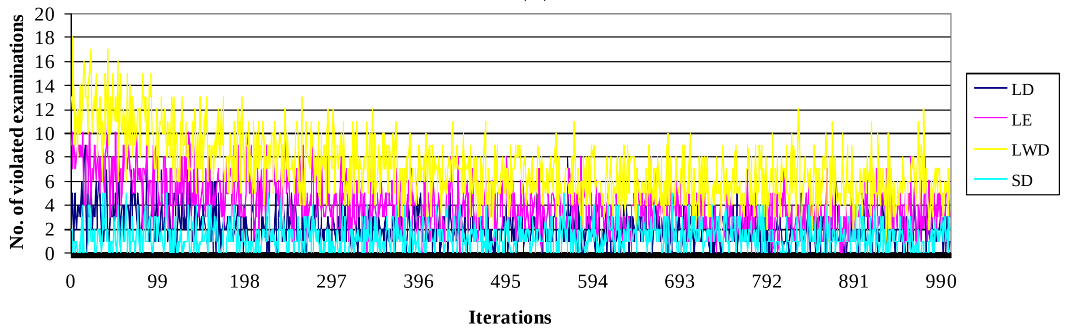


(d)

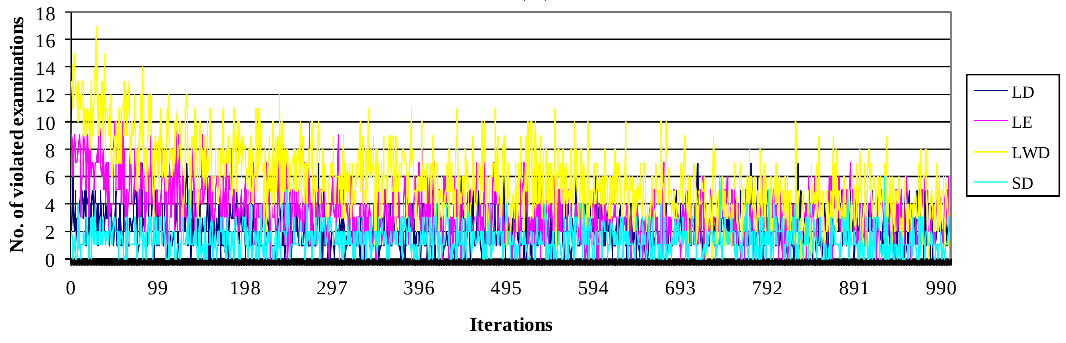
FIGURE A.2: The number of violated examinations at each iteration for car92 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



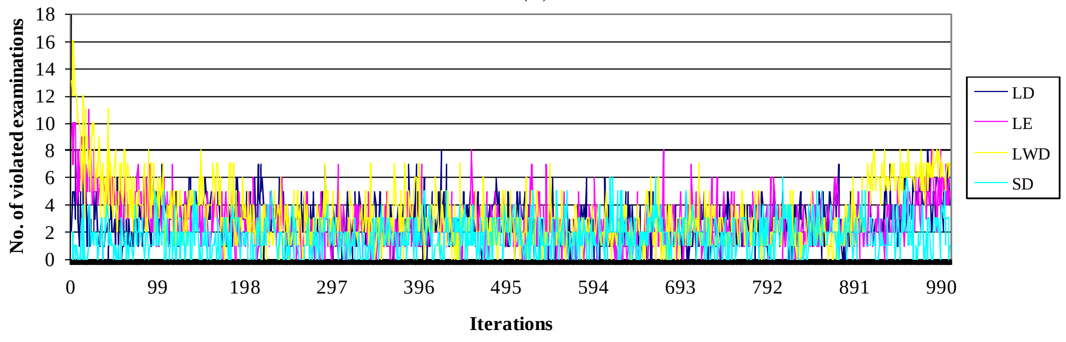
(a)



(b)

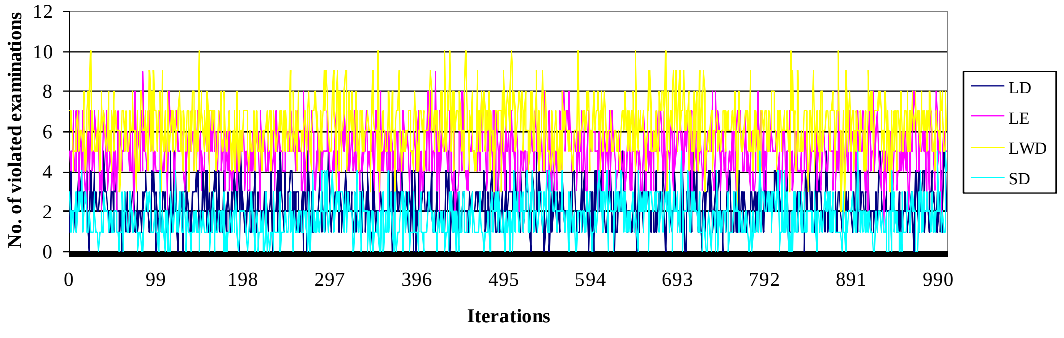


(c)

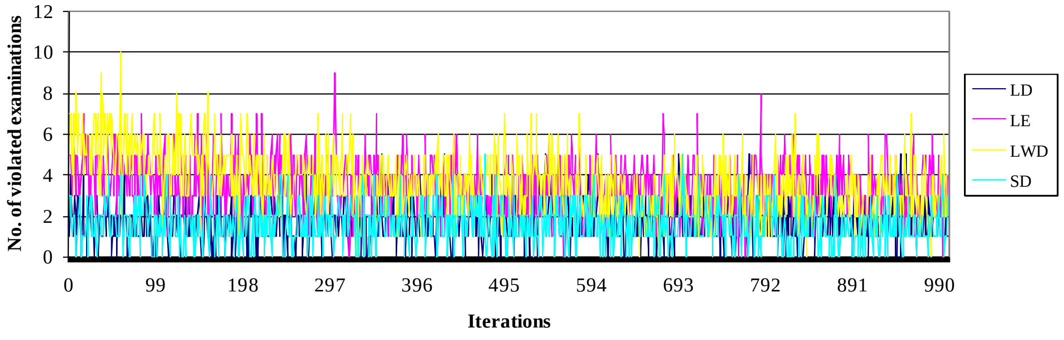


(d)

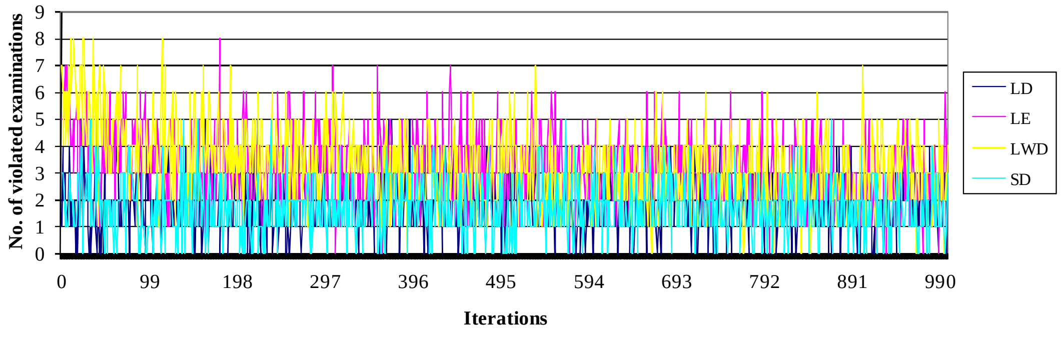
FIGURE A.3: The number of violated examinations at each iteration for ears83 I with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



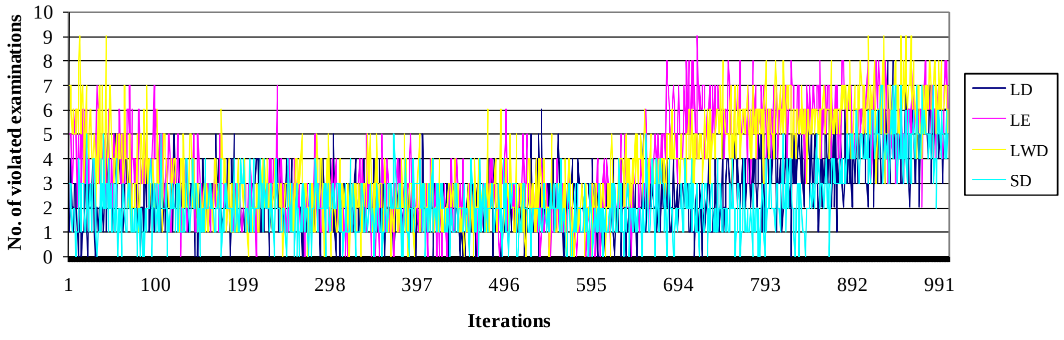
(a)



(b)

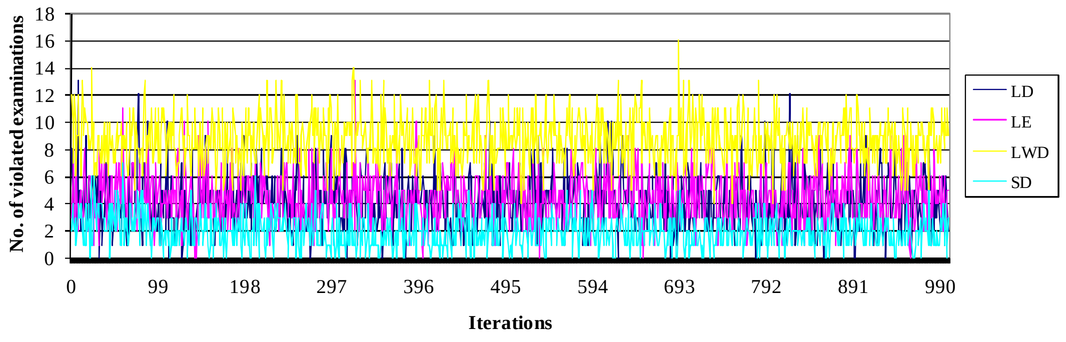


(c)

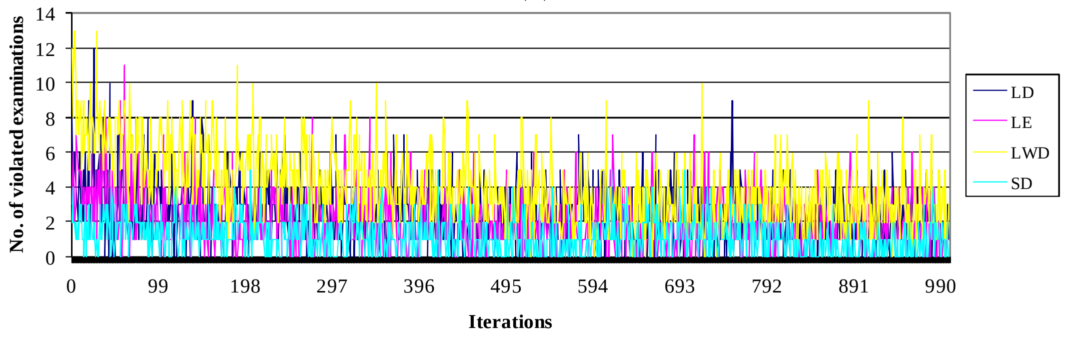


(d)

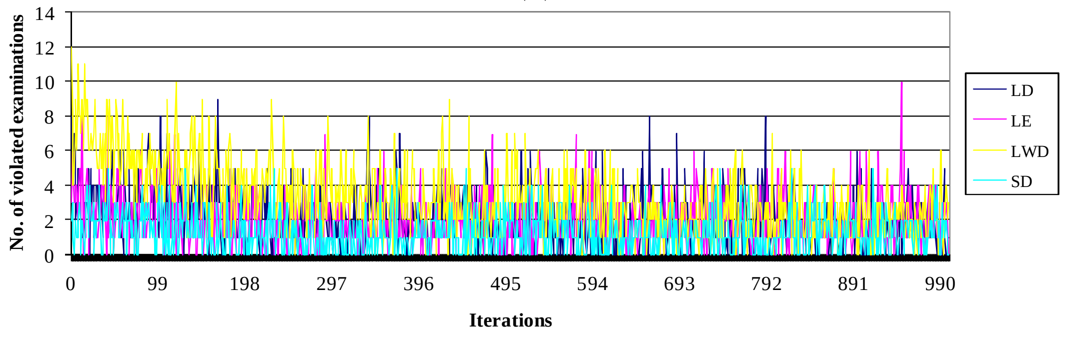
FIGURE A.4: The number of violated examinations at each iteration for hec92 I with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



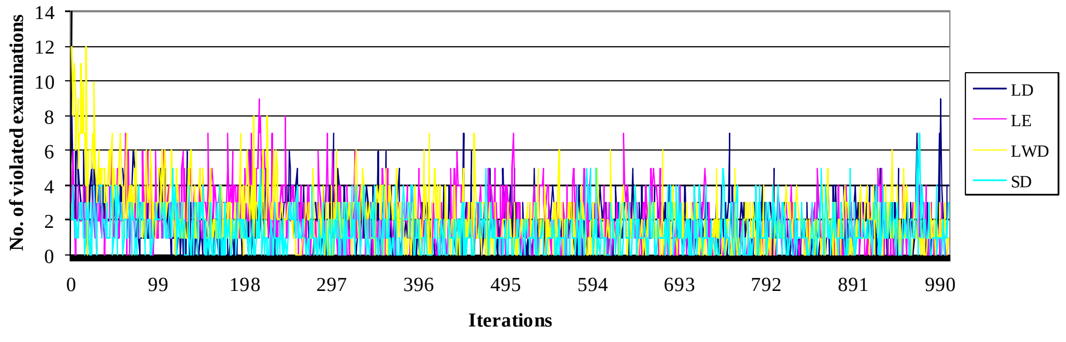
(a)



(b)

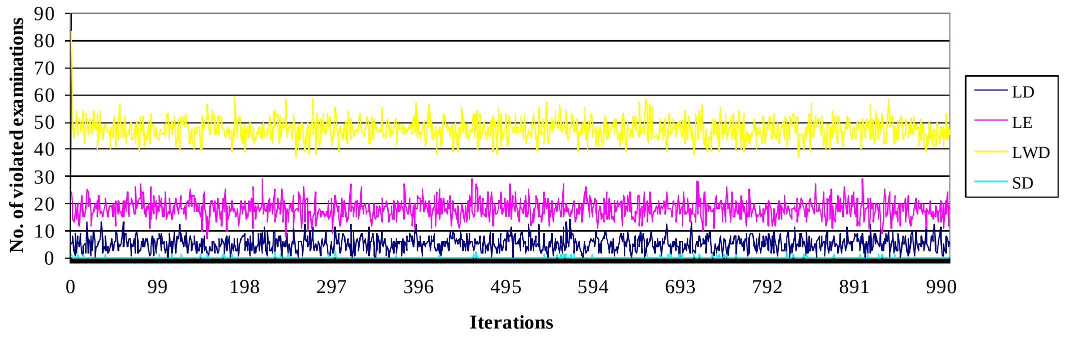


(c)

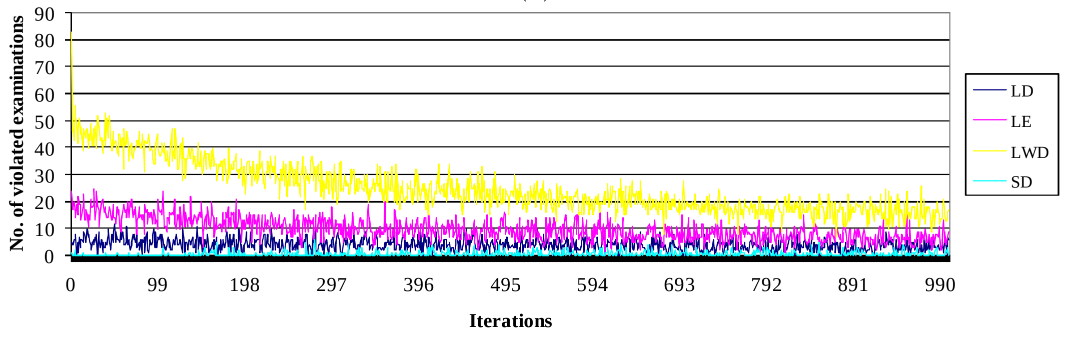


(d)

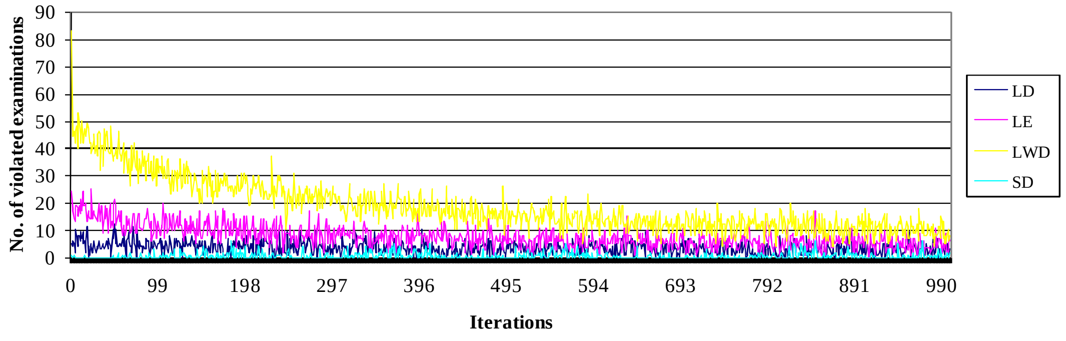
FIGURE A.5: The number of violated examinations at each iteration for kfu93 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



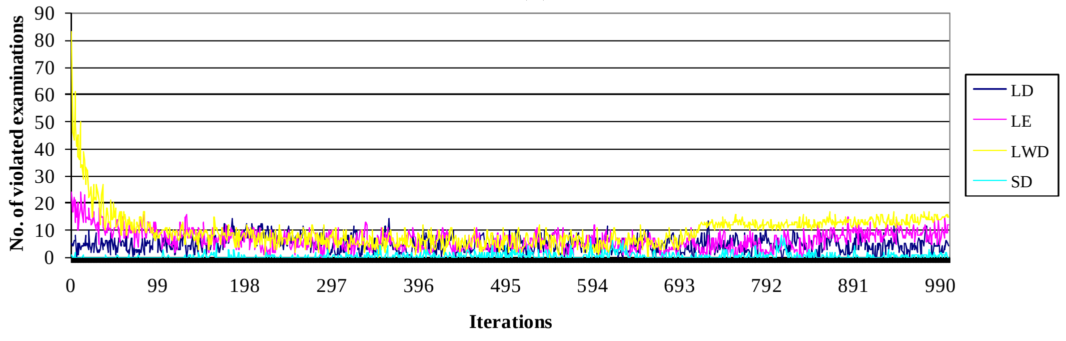
(a)



(b)

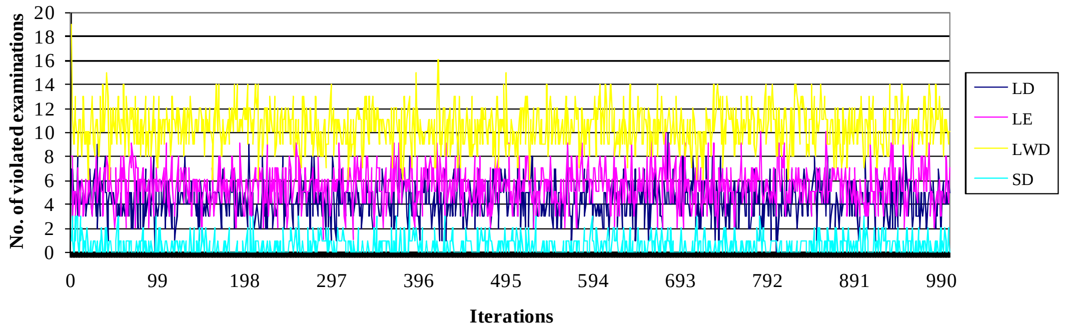


(c)

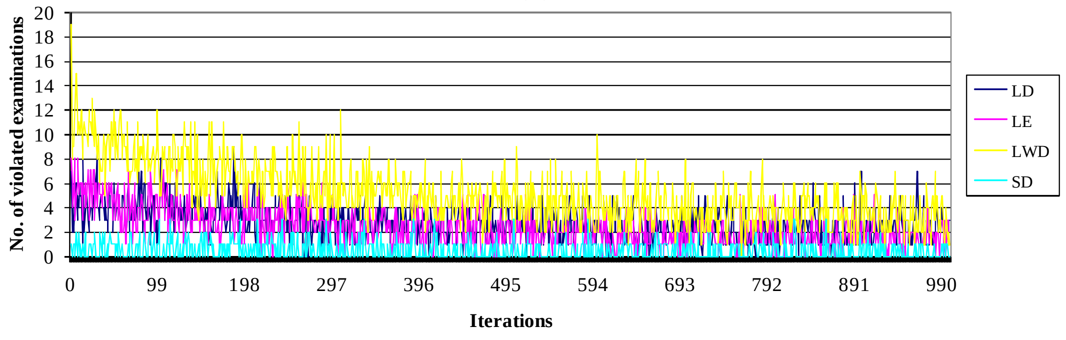


(d)

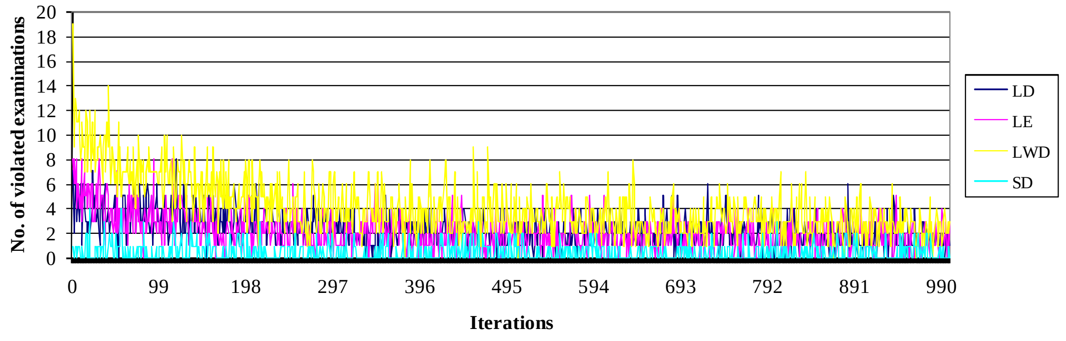
FIGURE A.6: The number of violated examinations at each iteration for pur93 I with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



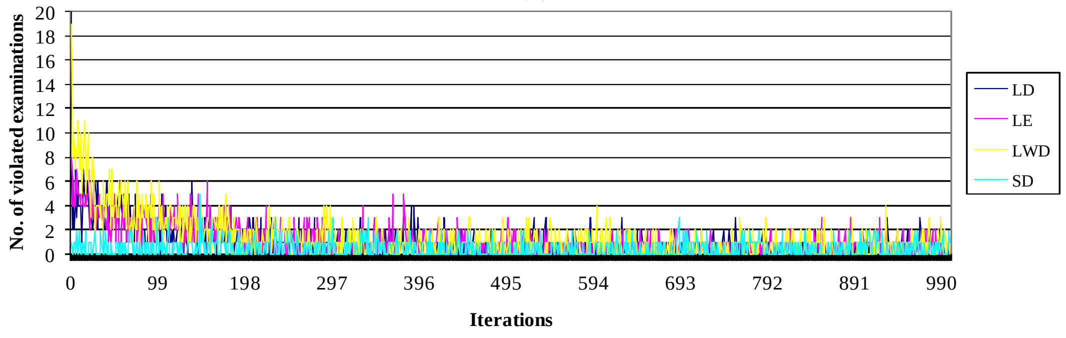
(a)



(b)

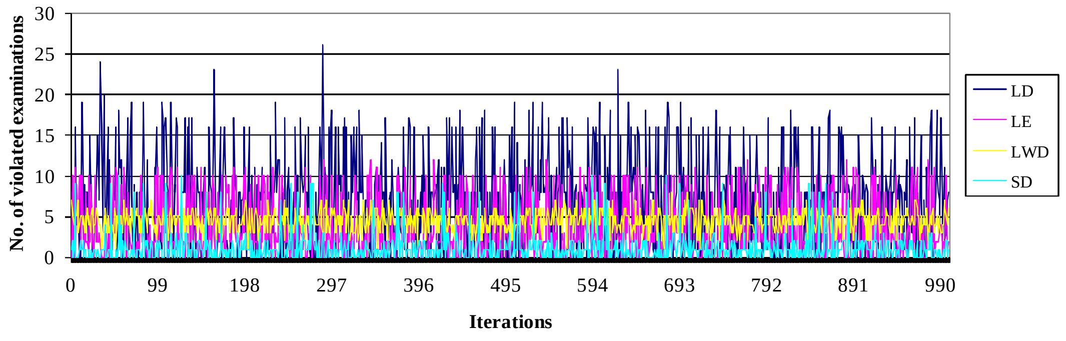


(c)

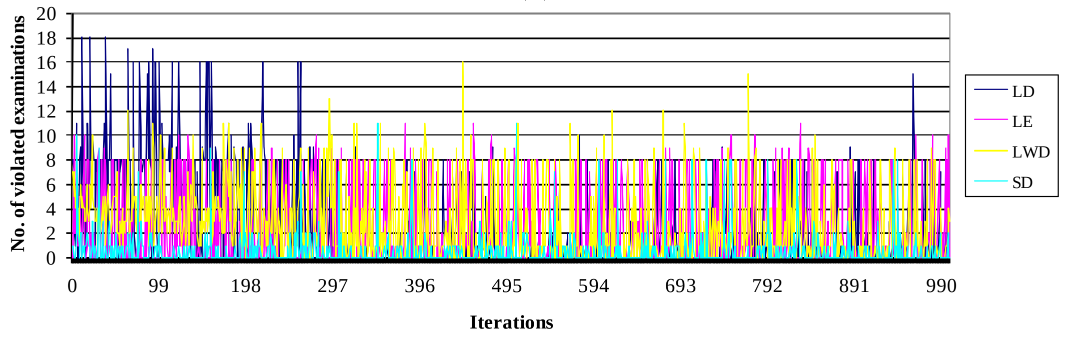


(d)

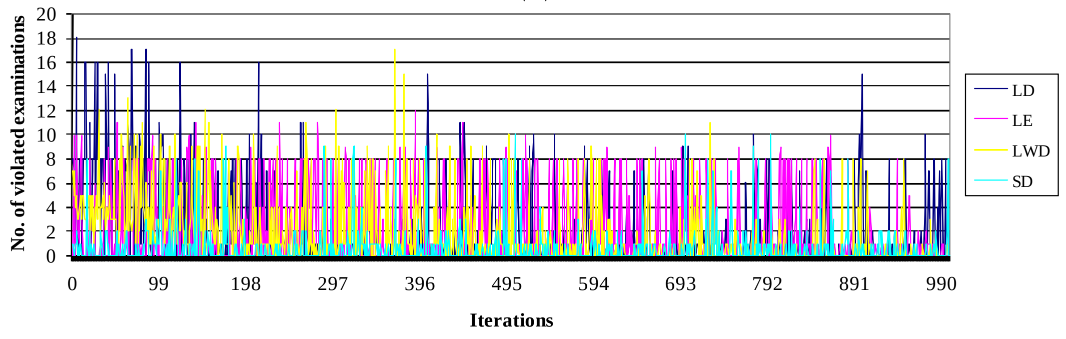
Figure A.7: The number of violated examinations at each iteration for rye92 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



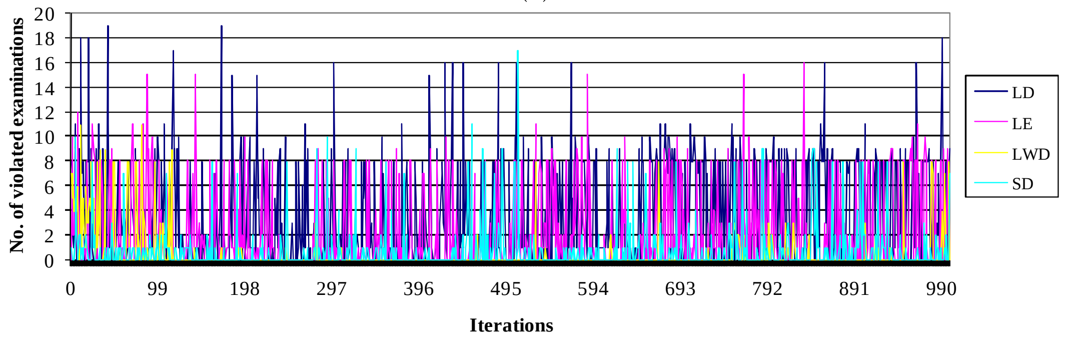
(a)



(b)

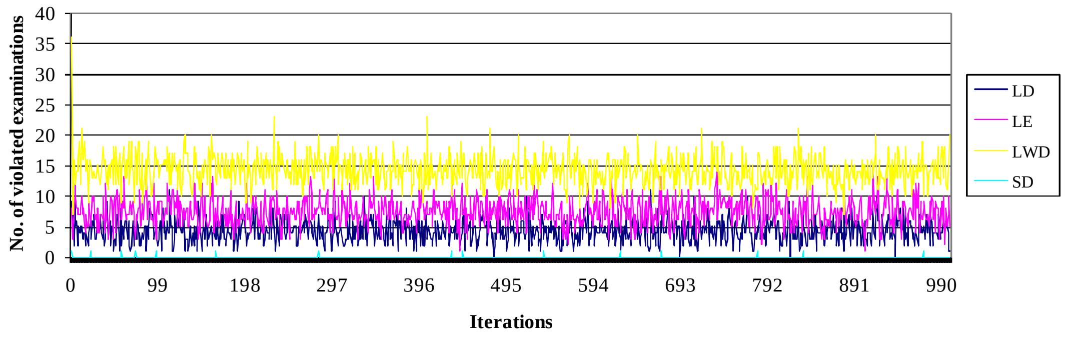


(c)

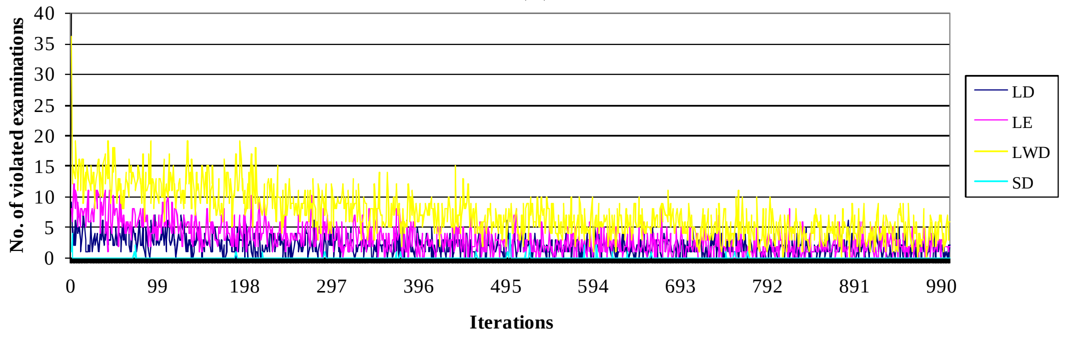


(d)

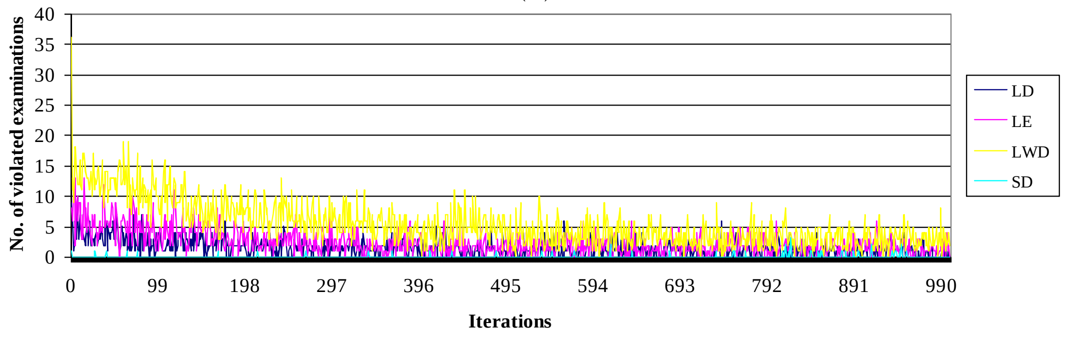
FIGURE A.8: The number of violated examinations at each iteration for sta83 I with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



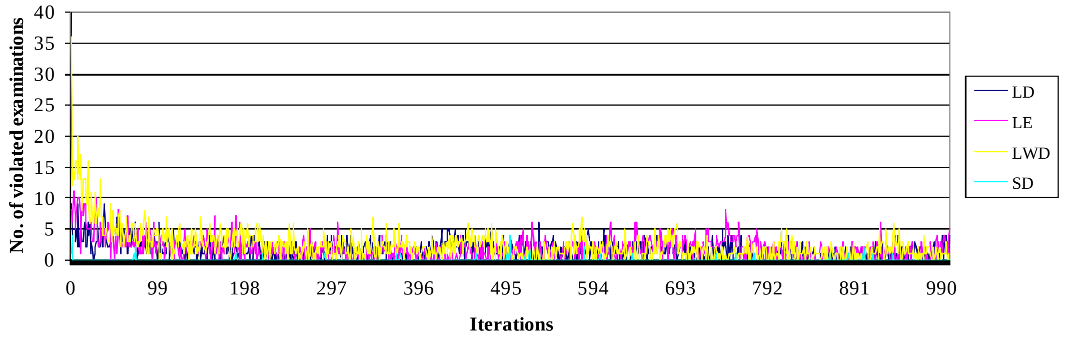
(a)



(b)

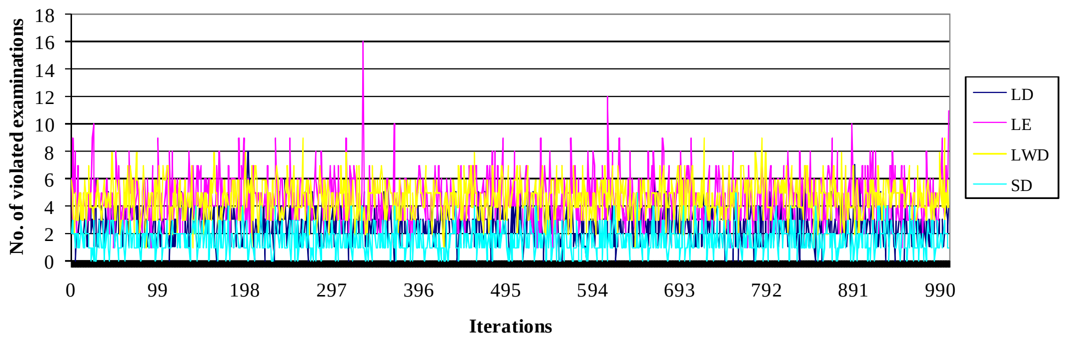


(c)

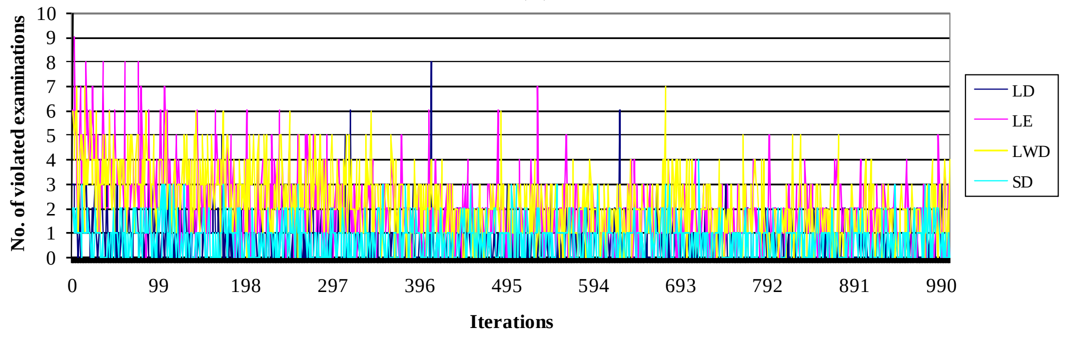


(d)

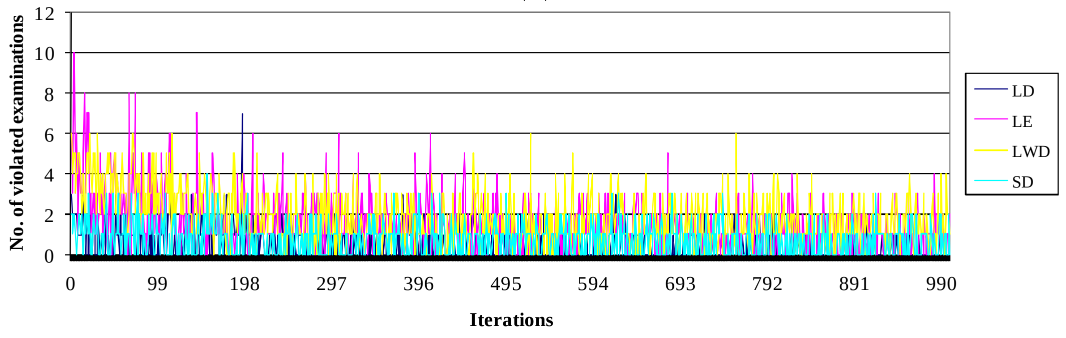
Figure A.9: The number of violated examinations at each iteration for uta92 I with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



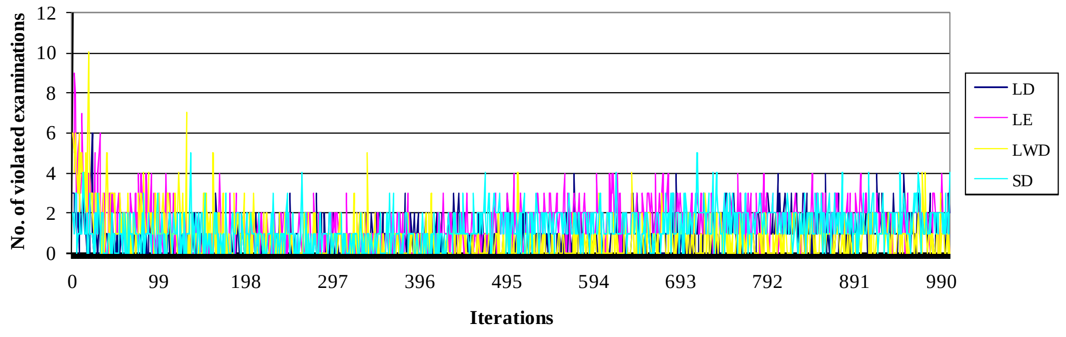
(a)



(b)

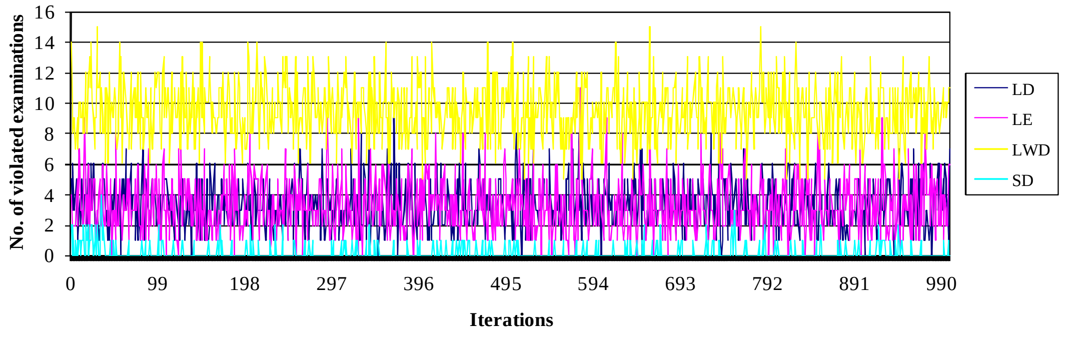


(c)

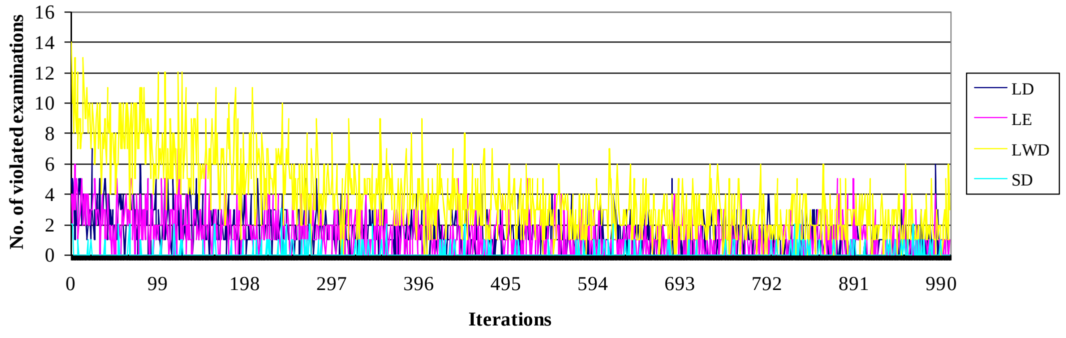


(d)

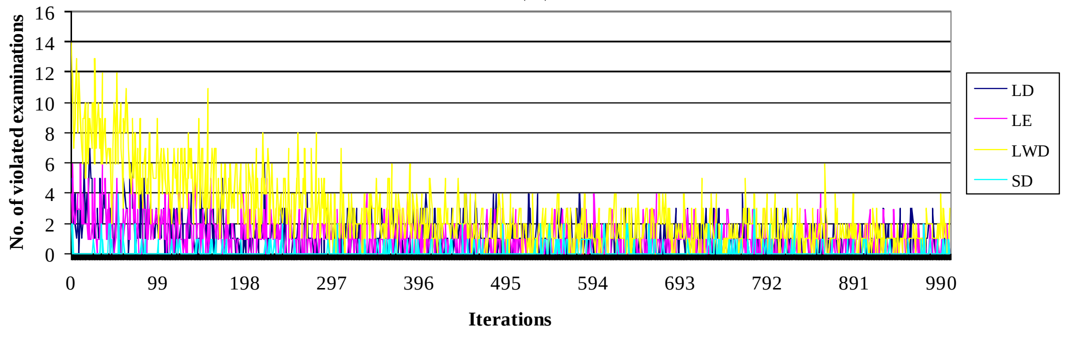
FIGURE A.10: The number of violated examinations at each iteration for ute92 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



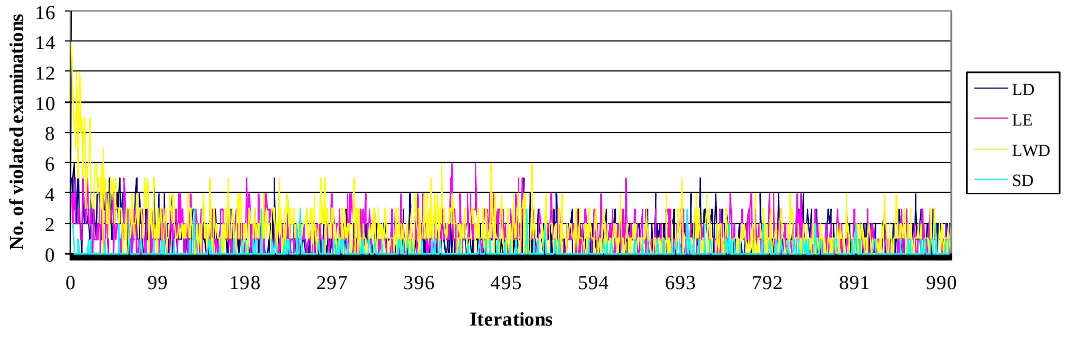
(a)



(b)

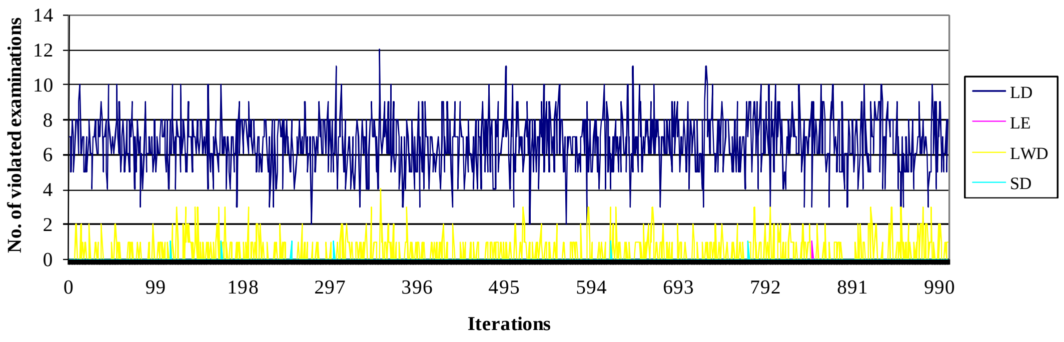


(c)

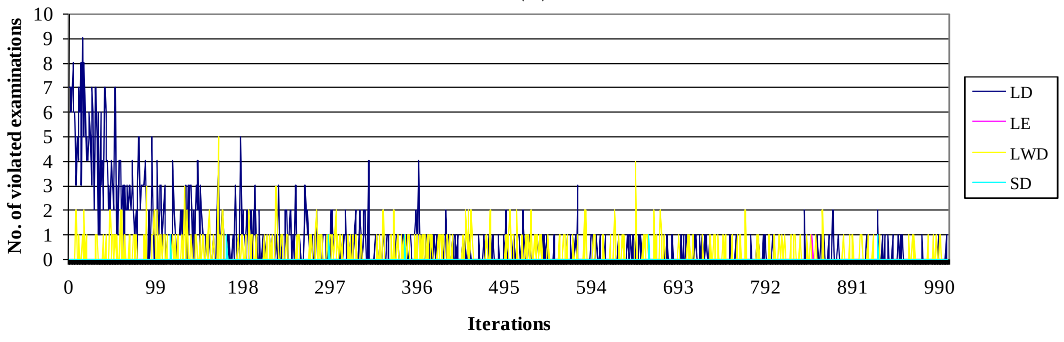


(d)

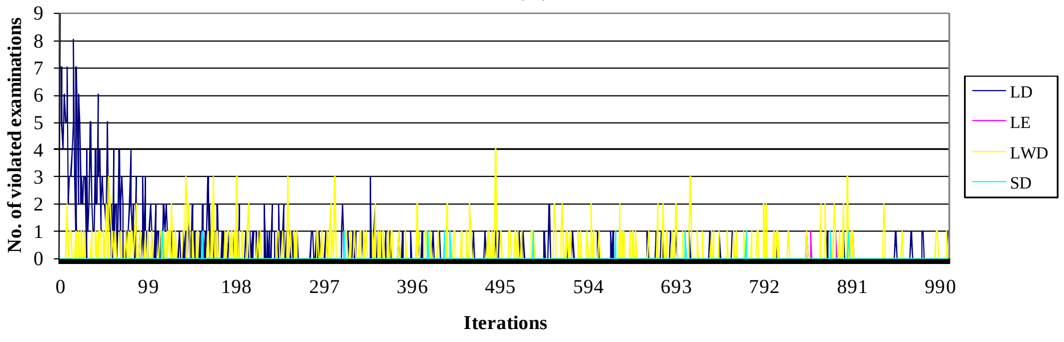
FIGURE A.11: The number of violated examinations at each iteration for tre92 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



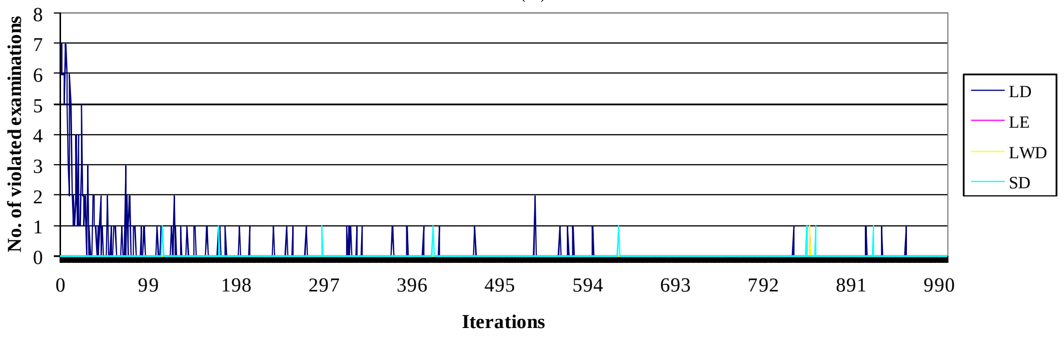
(a)



(b)

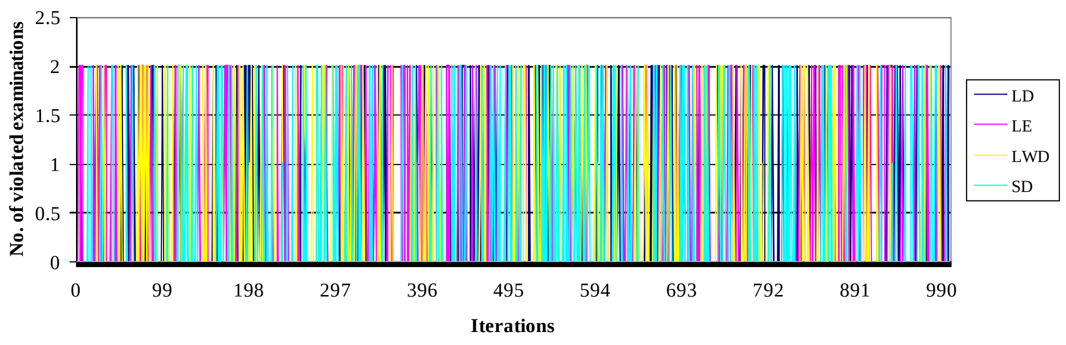


(c)

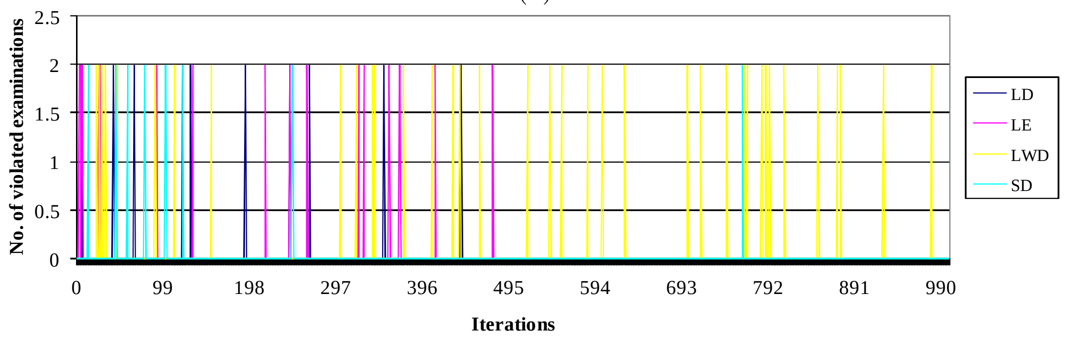


(d)

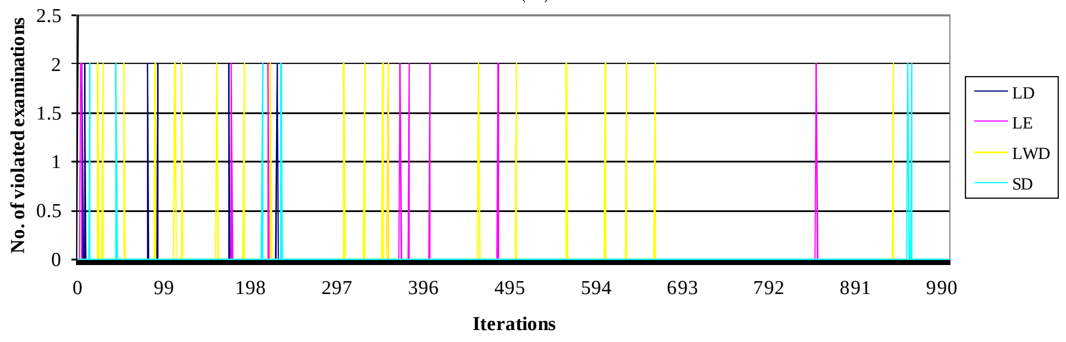
FIGURE A.12: The number of violated examinations at each iteration for Exam_1 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



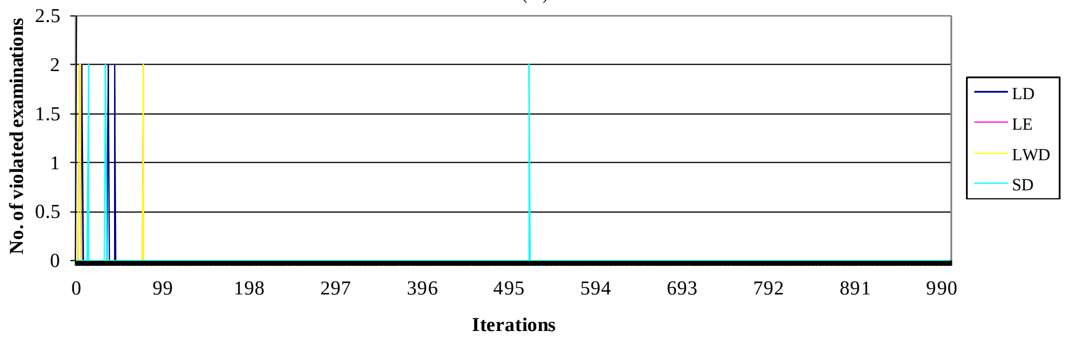
(a)



(b)

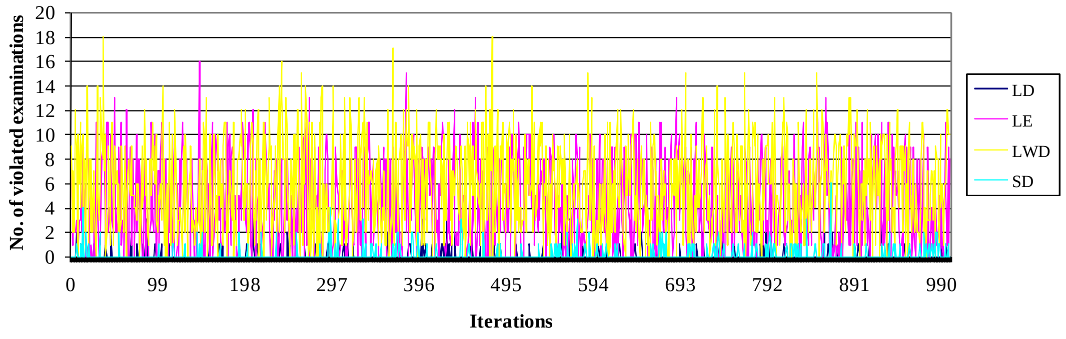


(c)

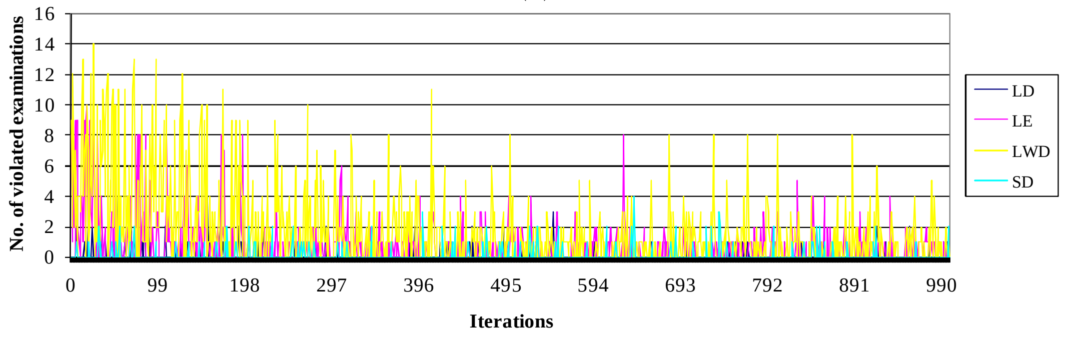


(d)

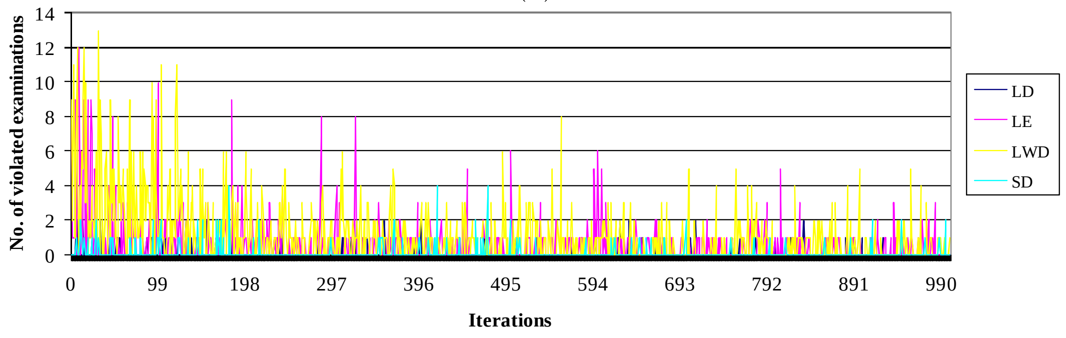
FIGURE A.13: The number of violated examinations at each iteration for Exam_2 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



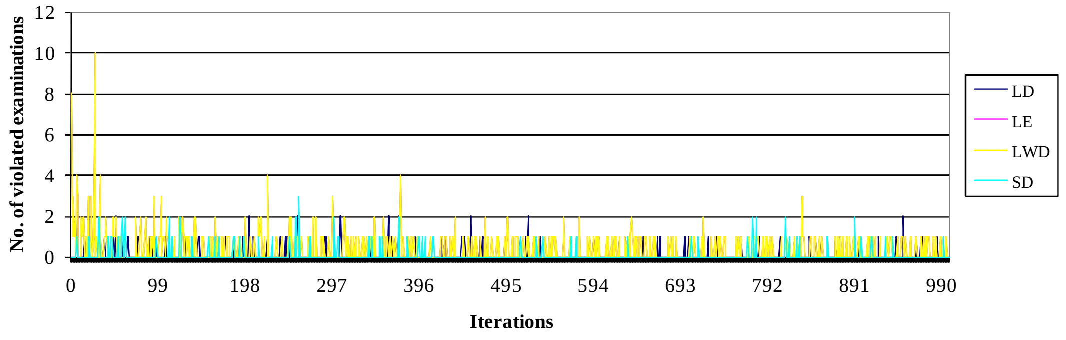
(a)



(b)

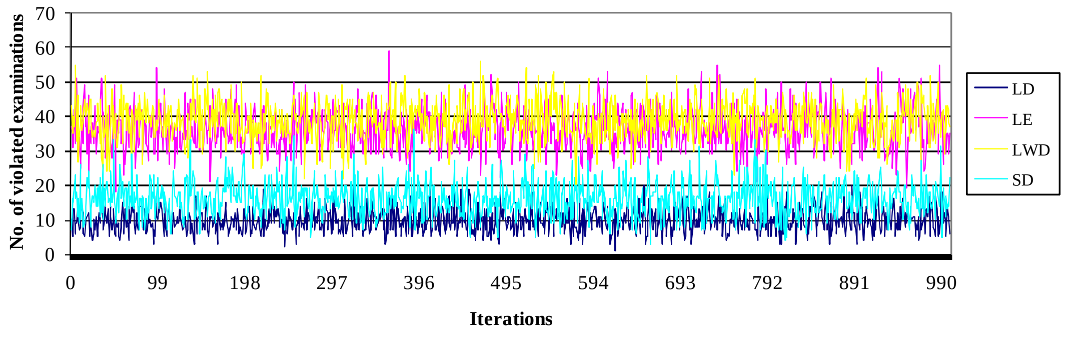


(c)

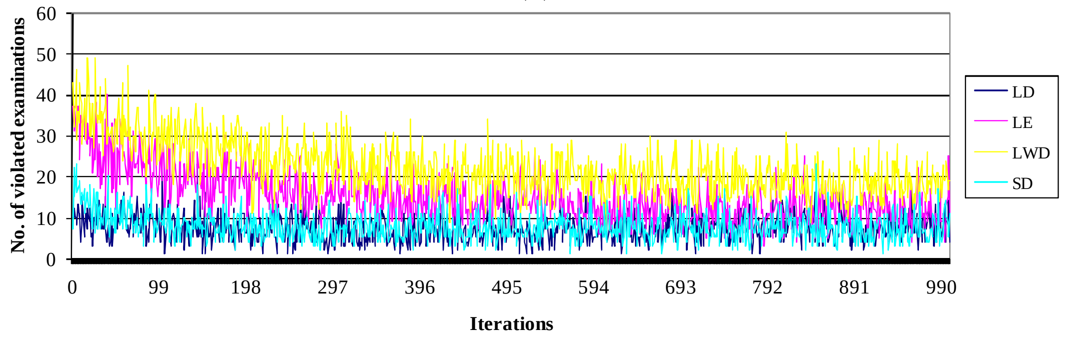


(d)

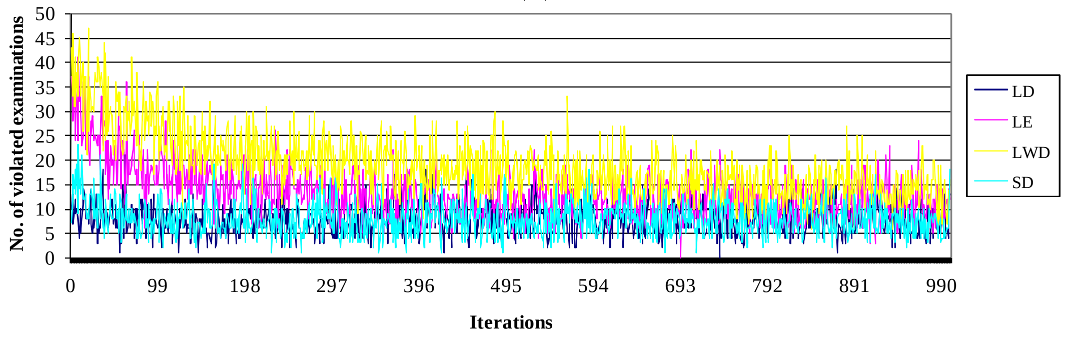
FIGURE A.14: The number of violated examinations at each iteration for Exam_3 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



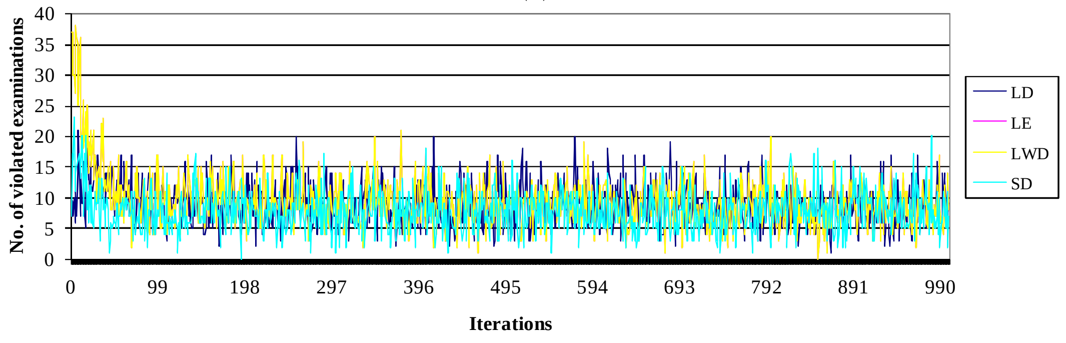
(a)



(b)

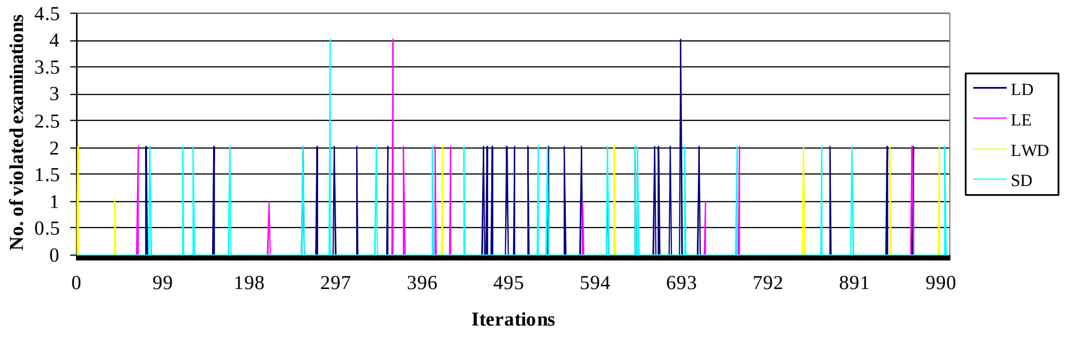


(c)

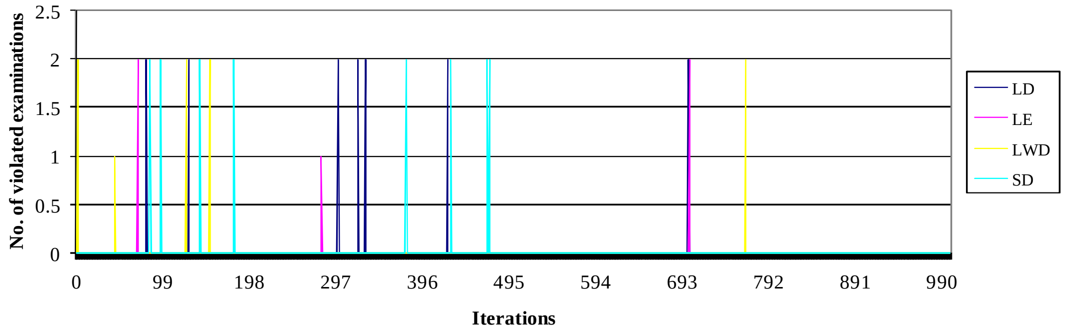


(d)

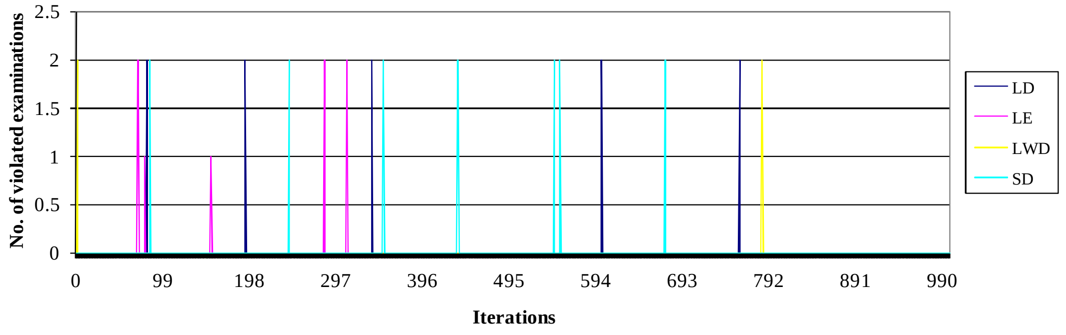
Figure A.15: The number of violated examinations at each iteration for Exam_4 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



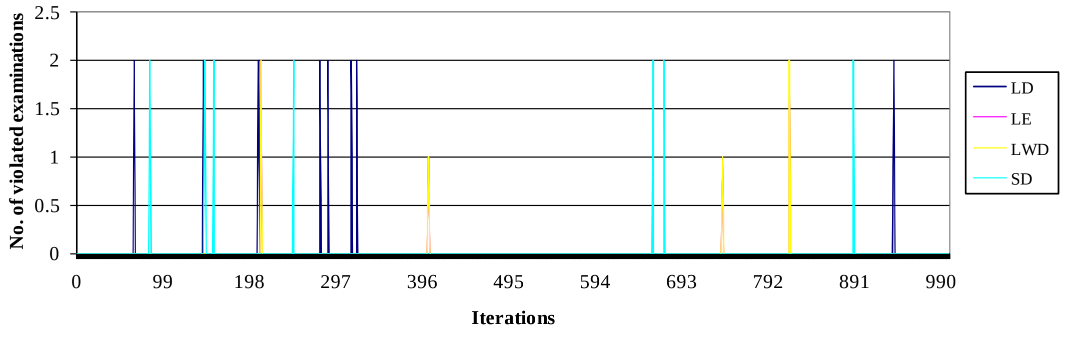
(a)



(b)

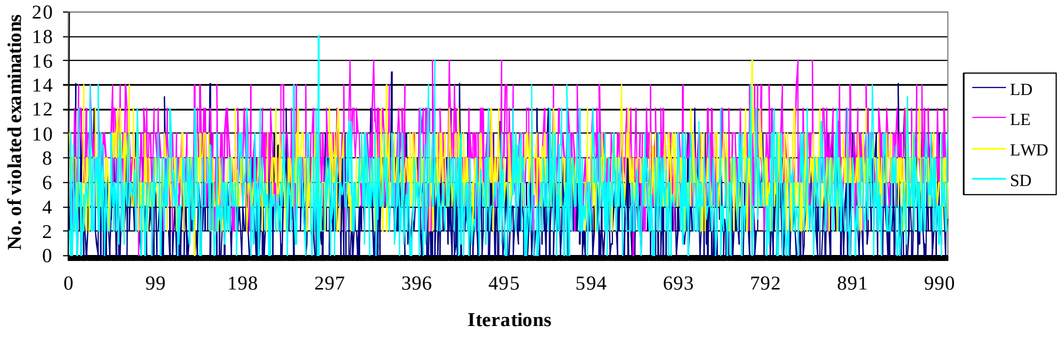


(c)

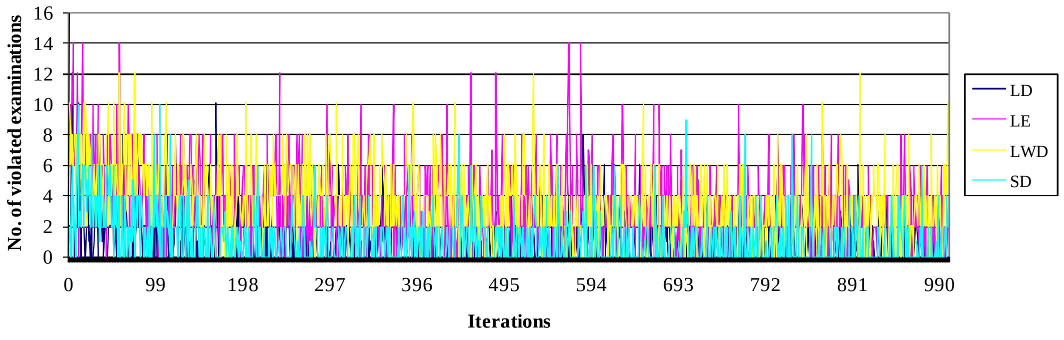


(d)

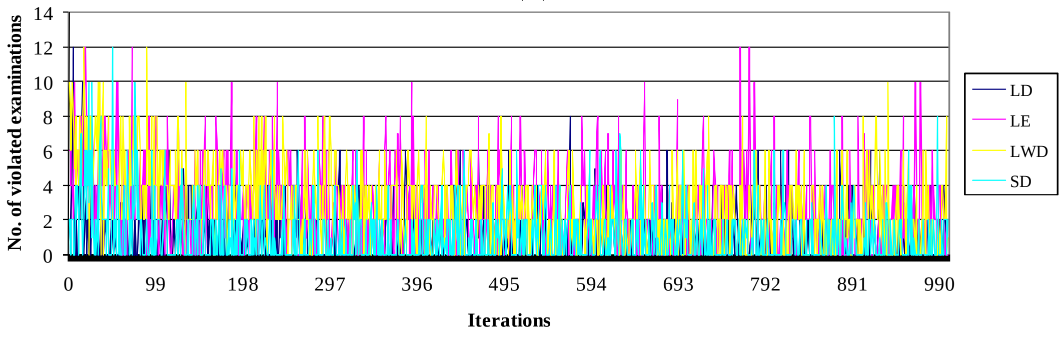
FIGURE A.16: The number of violated examinations at each iteration for Exam_5 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



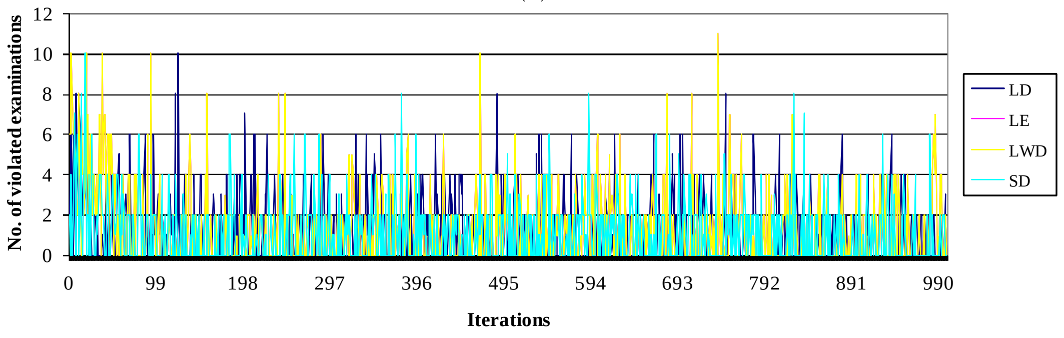
(a)



(b)

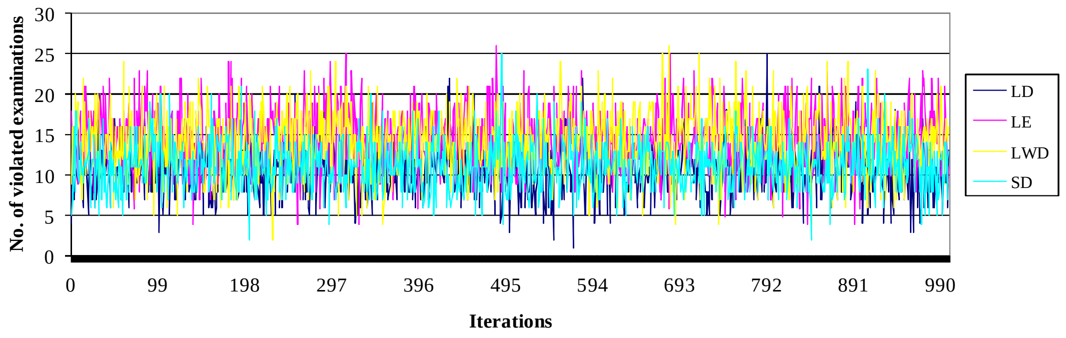


(c)

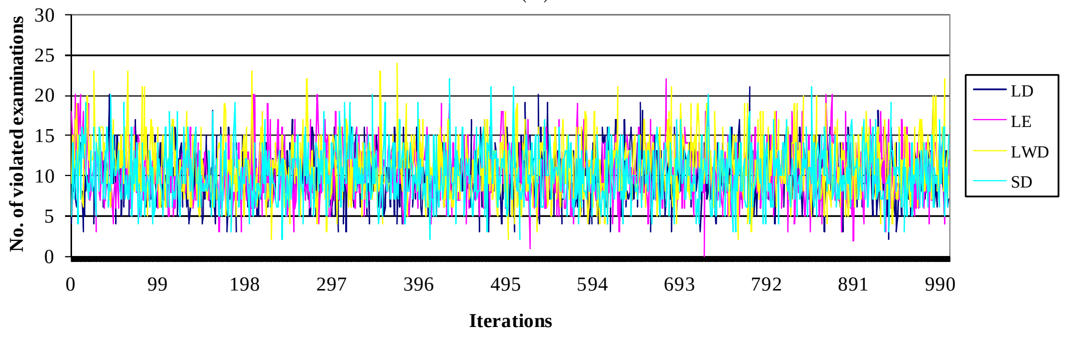


(d)

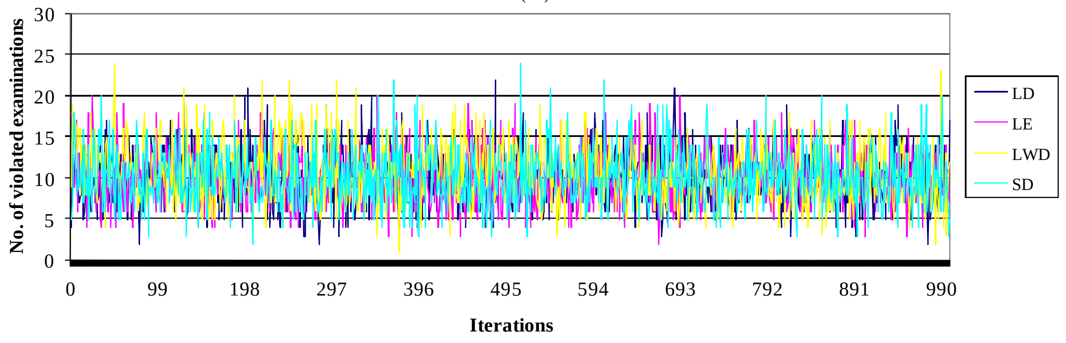
FIGURE A.17: The number of violated examinations at each iteration for Exam_6 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier



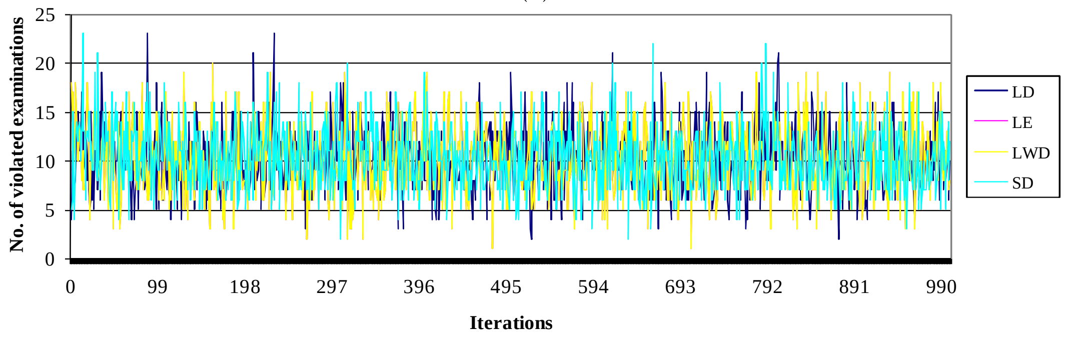
(a)



(b)



(c)



(d)

Figure A.18: The number of violated examinations at each iteration for Exam_12 with (a) custom (C), (b) additive (AD), multiplicative (MP) and exponential (EX) heuristic modifier

Bibliography

Aarts, E., Korst, J., and Michiels, W. (2005). *Search methodologies: Introductory tutorials in optimization anddecision support methodologies*, chapter 7: Simulated annealing, pages 187–210. Springer.

Abdul Rahim, S. K., Bargiela, A., and Qu, R. (2009). Granular modelling of exam to slot allocation. In *Proceedings of the 23rd European Conference on Modelling and Simulation (ECMS), Madrid, Spain*, pages 861–866.

Abdul Rahman, S., Bargiela, A., Burke, E. K., McCollum, B., and Özcan, E. (2009). Construction of examination timetables based on ordering heuristics. In *Proceedings of the 24th International Symposium on Computer and Information Sciences*, pages 727–732.

Abdul Rahman, S., Bargiela, A., Burke, E. K., McCollum, B., and Özcan, E. (2010). A construction approach for examination timetabling based on adaptive decomposition and ordering. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland*, pages 353–372.

Abdullah, S., Ahmadi, S., Burke, E. K., and Dror, M. (2007). Investigating ahuja-orlin's large neighbourhood search approach for examination timetabling. *OR Spectrum*, 29(2):351–372.

Abdullah, S., Burke, E. K., and McCollum, B. (2005). An investigation of variable neighbourhood search for university course timetabling. In *Proceedings of the 2nd Multi-disciplinary International Conference on Scheduling: Theory and Applications (MISTA), New York, USA, 18-21 July*, pages 413–427.

Abdullah, S., Shaker, K., McCollum, B., and McMullan, P. (2010). Incorporating great deluge with kempe chain neighbourhood structure for the enrolment-based course timetabling problem. In Yu, J., Greco, S., Lingras, P., Wang, G., and Skowron, A., editors, *Rough Set and Knowledge Technology*, volume 6401 of *Lecture Notes in Computer Science*, pages 70–77. Springer Berlin / Heidelberg.

Ahmadi, S., Barone, R., Cheng, P., Cowling, P., and McCollum, B. (2003). Perturbation based hyper-heuristic for examination timetabling problems. In *Proceeding of The 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), Nottingham.*

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

Al-Betar, M. A. and Khader, A. T. (2009). A hybrid harmony search for university course timetabling. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009), 10-12 August 2009, Dublin, Ireland.*

Al-Betar, M. A., Khader, A. T., and Gani, T. A. (2008). A harmony search algorithm for university course timetabling. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008.*

Al-Betar, M. A., Khader, A. T., and Thomas, J. J. (2010). A combination of metaheuristic components based on harmony search for the uncapacitated examination timetabling. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland.*

Alkan, A. and Özcan, E. (2003). Memetic algorithms for timetabling. In *Proceeding of IEEE Congress on Evolutionary Computation*, pages 1796–1802.

Asmuni, H., Burke, E. K., Garibaldi, J. M., and McCollum, B. (2005). Fuzzy multiple heuristic orderings for examination timetabling. In Burke, E. K. and Trick, M., editors, *Lecture notes in computer science: Practice and theory of automated timetabling V: selected papers from the 5th international conference*, volume 3616, pages 334–353. Berlin: Springer.

Asmuni, H., Burke, E. K., Garibaldi, J. M., and McCollum, B. (2007). A novel fuzzy approach to evaluate the quality of examination timetabling. In Burke, E. K. and Rudova, H., editors, *Lecture notes in computer science: Practice and theory of automated timetabling V: selected papers from the 6th international conference*, volume 3867, pages 327–346. Berlin: 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 and Operations Research*, 36(4):981–1001.

Atsuta, M., Nonobe, K., and Ibaraki, T. (2008). Itc2007 track 2: An approach using general csp solver. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008.* www.cs.qub.ac.uk/itc2007.

Avanthay, C., Hertz, A., and Zuffere, N. (2003). A variable neighbourhood search for grap coloring. *European Journal of Operational Research*, 151:379–388.

Avella, P., D'Auria, B., Salerno, S., and Vasil'ev, I. (2007). A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556.

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

Azimi, Z. N. (2004). Comparison of metaheuristic algorithms for examination timetabling problem. *Applied Mathematics and Computation*, 16(1-2):337–354.

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

Bardadym, V. A. (1996). Computer-aided school and university timetabling: The new wave. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 22–45, London, UK. Springer-Verlag.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematic*, 4:238–252.

Benoist, T., Gaudin, E., and Rottembourg, B. (2002). Constraint programming contribution to benders decomposition: A case study. In *Proceedings of the eigth international conference on principles and practice of constraint programming (CP), lecture notes in computer science*, volume 2470, pages 603–617.

Bilgin, B., Özcan, E., and Korkmaz, E. E. (2007). An experimental study on hyper-heuristics and exam scheduling. In Burke, E. K. and Rudova, H., editors, *Lecture notes in computer science (Practice and theory of automated timetabling VI: selected papers from the 6th international conference)*, volume 3867, pages 394–412. Berlin: Springer.

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

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

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

Bosch, R. and Trick, M. (2005). *Search Methodologies: Introductory tutorials in optimisation and decision support techniques*, chapter Integer Programming, pages 69–96. Berlin: Springer, Berlin.

Boufflet, J. P. and Nègre, S. (1996). Three methods used to solve an examination timetable problem. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 327–344, London, UK. Springer-Verlag.

Brandão de Oliveira, H. C. and Vasconcelos, G. C. (2010). A hybrid search method for the vehicle routing problem with time windows. *Annals of Operational Reseach*, 180:125–144.

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

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

Burke, E. K. and Barry, M., editors (2010). *Practice and theory of automated timetabling 2010*.

Burke, E. K. and Bykov, Y. (2006). Solving exam timetabling problems with the flex-deluge algorithm. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling (PATAT 2006)*, pages 370–372.

Burke, E. K. and Bykov, Y. (2008). A late acceptance strategy in hill-climbing for examination timetabling problems. In *Proceedings of the Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*.

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

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

Burke, E. K., Bykov, Y., and Petrovic, S. (2001). A multicriteria approach to examination timetabling. In Burke, E. K. and Erben, W., editors, *Lecture notes in computer science: Practice and theory of automated timetabling V: selected papers from the 3rd international conference*, volume 2079, pages 118–131, London, UK. Berlin: Springer.

Burke, E. K. and Carter, M. W., editors (1998). *Practice and theory of automated timetabling II*, volume 1408. Springer-Verlag.

Burke, E. K. and De Causmaecker, P., editors (2003). *Practice and theory of automated timetabling IV*, volume 2740. Springer-Verlag.

Burke, E. K., De Causmaecker, P., Petrovic, S., and Vanden Berghe, G. (2003b). *Metaheuristics: Computer decision-making*, chapter 7: Variable neighbourhood search for nurse rostering problems, pages 153–172. Kluwer.

Burke, E. K., Dror, M., Petrovic, S., and Qu, R. (2005a). Hybrid graph heuristics in hyper-heuristics applied to exam timetabling problems. In Golden, G. L., Raghavan, S., and Wasil, E. A., editors, *The next wave in computing, optimization, and decision technologies*, pages 79–91. Springer: Maryland.

Burke, E. K., Dror, M., Petrovic, S., and Qu, R. (2005b). *The next wave in computing, optimization, and decision technologies*, chapter Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems, pages 79–91. Springer.

Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., and Qu, R. (2004b). Analysing similarity in examination timetabling. In Burke, E. and Trick, M., editors, *Proceedings of The 5th International Conference on the Practice and Theory of Automated Timetabling. 18th-20th Aug, Pittsburgh, PA USA*, pages 89–106.

Burke, E. K., Eckersley, A. J., 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. K., Elliman, D., Ford, P. H., and Weare, R. F. (1996a). Examination timetabling in british universities: A survey. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 76–90, London, UK. Springer-Verlag.

Burke, E. K., Elliman, D., and Weare, R. F. (1995). Specialised recombinative operators for timetabling problems. In *Lecture notes in computer science: Selected Papers from Artificial Intelligence and Simulation of Behaviour (AISB) Workshop on Evolutionary Computing*, volume 993, pages 75–85, London, UK. Springer-Verlag.

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

Burke, E. K., Elliman, D. G., and Weare, R. (1994). 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., editors (2001). *Practice and theory of automated timetabling III*, volume 2079. Springer-Verlag.

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

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., and Özcan, E. (2010b). *Handbook of metaheuristics*, volume 146, chapter A classification of hyper-heuristic approaches, pages 449–468. Springer.

Burke, E. K. and Kendall, G., editors (2005). *Search methodologies: Introductory tutorials in optimization and decision support techniques.* Springer.

Burke, E. K., Kendall, G., Misir, M., and Özcan, E. (2010c). Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research.*

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., Kingston, J., and de Werra, D. (2004c). *Handbook of graph theory*, chapter Applications to timetabling, pages 445–474. Chapman Hall/CRC Press.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010d). Decomposition, reformulation, and diving in university course timetabling. *Computers & Operations Research*, 37(1):582–597.

Burke, E. K., McCollum, B., McMullan, P., and Parkes, A. J. (2008). Multi-objective aspects of the examination timetabling competition track. In *Procedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT 2008.*

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

Burke, E. K., Newall, J., and Weare, R. F. (1998a). A simple heuristically guided search for the timetable problem. In Alpaydin, E. and Fyte, C., editors, *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*, pages 574–579, University of La Laguna, Spain 1998. ICSC Academic Press.

Burke, E. K. and Newall, J. P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE Trans. Evolutionary Computation*, 3(1):63–74.

Burke, E. K. and Newall, J. P. (2003). Enhancing timetable solutions with local search methods. In *Lecture notes in computer science: Practice and theory of automated timetabling IV: selected papers from the 4th international conference*, volume 2740, pages 195–206. Springer-Verlag.

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

Burke, E. K., Newall, J. P., and Weare, R. F. (1996b). A memetic algorithm for university exam timetabling. In Burke, E. K. and Ross, P., editors, *Lecture notes in computer science: Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, volume 1153, pages 241–250, London, UK. Springer-Verlag.

Burke, E. K., Newall, J. P., and Weare, R. F. (1998b). Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation Journal (special issue on Scheduling)*, 6(1):81–103.

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

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. (2010e). Linear combinations of heuristics for examination timetabling. *Annals of Operations Research.*

Burke, E. K., Qu, R., and Soghier, A. (2009). Adaptive selection of heuristics within a grasp for exam timetabling problems. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009), 10-12 August 2009, Dublin, Ireland*, pages 409–423.

Burke, E. K., Qu, R., and Soghier, A. (2010f). Adaptive selection of heuristics for improving constructed exam timetables. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland*, pages 136–152.

Burke, E. K. and Ross, P., editors (1996). *Practice and theory of automated timetabling*, volume 1153. Springer-Verlag.

Burke, E. K. and Rudová, H., editors (2007). *Practice and theory of automated timetabling VI*, volume 3667. Springer-Verlag.

Burke, E. K. and Trick, M., editors (2005). *Practice and theory of automated timetabling V*, volume 3616. Springer-Verlag.

Caporossi, G. and Hansen, P. (2000). Variable neighbourhood search for external graphs: I the autographix system. *Discrete Mathematics*, 212:29–44.

Caramia, M., Dell'Olmo, P., and Italiano, G. (2008). Novel local search based approaches to university examination timetabling. *INFORMS Journal of Computing*, 20:86–99.

Carrington, J. R., Pham, N., Qu, R., and Yellen, J. (2007). An enhanced weighted graph model for examination/course timetabling. In *Proceedings of the 26th Workshop of the UK Planning and Scheduling (PlanSIG2007).*

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Operational Research*, 34(2):193–202.

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

Carter, M. W. and Laporte, G. (1996). Recent developments in practical examination timetabling. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 3–21, London, UK. Springer-Verlag.

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. (1996). Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 47(3):373–383.

Casey, S. and Thompson, J. (2003). Grasping the examination scheduling problem. In *Lecture notes in computer science: Practice and theory of automated timetabling IV: selected papers from the 4th international conference*, volume 2740, pages 234–244. Berlin: Springer.

Chen, X. and Bushnell, M. L. (1996). *Efficient branch and bound search with application to computer-aided design.* Kluwer Academic Publishers.

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 *Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)*, pages 165–172.

Cole, A. J. (1964). The preparation of examination timetables using a smallstore computer. *Computer Journal*, 7:117–121.

Corne, D., Ross, P., and Fang, H. (1994). Evolutionary timetabling: Practice, prospects and work in progress. In Prosser, P., editor, *Proceedings of the UK Planning and Scheduling SIG Workshop.*

Corr, P., McCollum, B., McGreevy, M., and McMullan, P. (2006). A new neural network based construction heuristic for the examination timetabling problem. In *Proceedings of the Parallel Problem Solving from Nature - PPSN IX, Lecture Notes in Computer Science*, volume 4193, pages 392–401. Springer-Verlag.

Côté, P., Wong, T., and Sabourin, R. (2005). A hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem. In *Lecture Notes in Computer Science: Selected papers from the 5th International Conference on Practice and Theory of Automated Timetabling (PATAT 2004)*, volume 3616, pages 151–168.

De Causmaecker, P., Demeester, P., and Vanden Berghe, G. (2009). A decomposed metaheuristic approach for a real-world university timetabling problem. *European Journal of Operational Research*, 195:307–318.

De Smet, G. (2008). Itc2007 - examination track. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008.*

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19(2):151–162.

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

Detienne, B., Péridy, L. andPinson, E., and Rivreau, D. (2009). Cut generation for an employee timetabling problem. *European Journal of Operational Research*, 197(3):1178–1184.

Di Gaspero, L. (2002). Recolour, shake and kick: A recipe for the examination timetabling problem. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium.*

Di Gaspero, L. and Schaerf, A. (2001). Tabu search techniques for examination timetabling. In *Lecture notes in computer science: Practice and theory of automated timetabling III: selected papers from the 3rd international conference*, pages 104–117, London, UK. Berlin: Springer.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344:243–278.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant sytem: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 26 (1):29–41.

Dowsland, K. A. and Thompson, J. (2005). Ant colony optimization for the examination scheduling problem. *Journal of Operational Research Society*, 56:426–438.

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

Duong, T. A. and Lam, K. H. (2004). Combining constraint programming and simulated annealing on university exam timetabling. In *Proceedings of the RIVF 2004 conference, Hanoi, Vietnam.*

Eley, M. (2007). Ant algorithms for the exam timetabling problem. In Burke, E. K. and Rudova, H., editors, *Lecture notes in computer science: Practice and theory of automated timetabling VI: selected papers from the 6th international conference*, volume 3867, pages 364–382. Berlin: Springer.

Erben, W. (2001). A grouping genetic algorithm for graph colouring and exam timetabling. In Burke, E. K. and De Causmaecker, P., editors, *Lecture Notes in Computer Science, The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT IV)*, volume 2740, pages 132–156. Berlin: Springer.

Ergül, A. (1996). Ga-based examination scheduling experience at middle east technical university. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 212–226, London, UK. Springer-Verlag.

Ersoy, E., Özcan, E., and Sima Uyar, A. (2007). Memetic algorithms and hyperhill-climbers. In Baptiste, P.and Kendall, G., Kordon, A. M., and Sourd, F., editors, *Lecture notes in computer science: Multidisciplinary international conference on scheduling: theory and applications: selected papers from the 3rd international conference*, pages 159–166.

Even, S., Itai, A., and Shamir, A. (1976). On the complexity of timetable and multi-commodity flow problems. *SIAM Journal on Computing*, 5(4):691–703.

Fleszar, K. and Hindi, K. H. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155(2):402–413.

Geem, Z. W., Kim, J. H., and Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, 76 (2):60–68.

Gendreau, M. and Potvin, J. (2005). *Introductory tutorials in optimisation, decision support and search methodology*, chapter 6: Tabu search, pages 165–186. Kluwer.

Gersmann, K. and Hammer, B. (2003). Improving iterative repair strategies for scheduling with the svm. In *Proceedings of european symposium on artificial neural networks.*

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computer and Operational Research*, 13 (5):533–549.

Glover, F. (1989). Tabu search-part i. *ORSA Journal on Computing*, 1 (3):190–206.

Glover, F. (1990). Tabu search-part ii. *ORSA Journal on Computing*, 2 (1):4–32.

Glover, F. and Kochenberher, G. (2003). *Handbook of meta-heuristics.* Kluwer.

Glover, F. and Laguna, M. (1997). *Tabu search.* Kluwer Academic Publisher.

Gogos, C., Alefragis, P., and Housos, E. (2008). A multi-staged algorithmic process for the solution of the examination timetabling problem. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008.*

Gogos, C., Alefragis, P., and Housos, E. (2010a). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operation Research*, 3:1–3.

Gogos, C., Goulas, G., Alefragis, P., and Housos, E. (2009). Pursuits of better results for the examination timetabling problem using grid resources. In *Proceedings of the Computational intelligence in scheduling (CISched09).*

Gogos, C., Goulas, G., Alefragis, P., Kolonias, V., and Housos, E. (2010b). Distributed scatter search for the examination timetabling problem. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland*, pages 211–223.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Hansen, P., Mladenović, N., and Perez-Britos, D. (2001). Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350.

Haykin, S. (1999). *Neural networks: A comprehensive foundation.* Prentice Hall.

Ibaraki, T. (2008). Problem solving by general purpose solvers. In *Proceeding of the 7th International Symposium of Operations Research and Its Applications (ISORA08), Lijian, China*, pages 10–17.

Ibaraki, T. (2010). A personal perspective on problem solving by general purpose solvers. *International Transactions in Operational Research*, 17:303–315.

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*, 10:353–37.

Kahar, M. N. 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*, 207:557–565.

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.

Kendall, G. and Mohd Hussin, N. (2005a). An investigation of a tabu search based hyper-heuristic for examination timetabling. In Kendall, G., Burke, E. K., and Petrovic, S., editors, *Selected papers from Multidisciplinary Scheduling: Theory and Applications 1st International Conference (MISTA2003).* Springer.

Kendall, G. and Mohd Hussin, N. (2005b). Tabu search hyper-heuristic approach to the examination timetabling problem at university technology mara. In Burke, E. K. and Trick, M., editors, *Lecture notes in computer science. Practice and theory of automated timetabling V: selected papers from the 5th international conference*, pages 199–217, Pittsburgh, USA.

Kiaer, L. and Yellen, J. (1992). Weighted graphs and university course timetabling. *Computers and Operations Research*, 19(1):59–67.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimisation by simulated annealing. *Science*, 220:671–380.

Landa-Silva, D. and Obit, J. H. (2008). Great deluge with nonlinear decay rate for solving course timetabling problems. In *Proceedings of the 2008 IEEE Conference on Intelligent Systems (IS 2008)*, pages 8.11–8.18. IEEE press.

Landa-Silva, J. D., Burke, E. K., and Petrovic, S. (2004). *Metaheuristic for multiobjective optimisation, lecture notes in economics and mathematical systems*, volume 535, chapter An introduction to multiobjective metaheuristics for scheduling and timetabling, pages 91–129. Springer.

Laporte, G. and Desroches, S. (1984). Examination timetabling by computer. *Computers & Operations Research*, 11(4):351–360.

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30(1):167–190.

Martinez, Y., Wauters, T., De Causmaecker, P., Nowe, A., Verbeeck, K., Bello, R., and Suarez, J. (2010). Reinforcement learning approaches for the parallel machines job shop scheduling problem. In *Proceedings of the cubaflanders workshop on machine learning and knowledge discovery.*

McCollum, B. (2007). A perspective on bridging the gap between research and practice in university timetabling. In Burke, E. K. and Rudova, H., editors, *Proceedings of the Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science*, volume 3867, pages 3–23. Springer.

McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., and Qu, R. (2008). A new model for automated examination timetabling. *Annals of OR*, 2:2–3.

McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., and Abdullah, S. (2009). An extended great deluge approach to the examination timetabling problem. In *Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA09), Dublin.*

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., and Burke, E. K. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130.

Mehta, N. K. (1981). The application of a graph coloring method to an examination scheduling problem. *INTERFACES*, 11(5):57–65.

Meisels, A., Ell-sana, J., and Gudes, E. (1994). Decomposing and solving timetabling constraint networks. *Computational Intelligence*, 1997:486–505.

Meisels, A. and Schaerf, A. (2003). Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, 39 (1-2):41–59.

Mercier, A., Cordeau, J.-E., and Soumis, F. (2005). A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451–1476.

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. K. and Erben, W., editors, *Lecture notes in computer science: Practice and theory of automated timetabling V: selected papers from the 4th international conference*, volume 2740, pages 207–231. Berlin: Springer.

Meyers, C. and Orlin, J. B. (2007). Very large-scale neighborhood search techniques in timetabling problems. In Burke, E. K. and Rudova, H., editors, *Lecture notes in computer science: Practice and theory of automated timetabling VI: selected papers from the 6th international conference*, volume 3867, pages 24–39. Berlin: Springer.

Mihaylov, M., Le Borgne, Y., Nowe, A., and Tuyls, K. (2010). Decentralized reinforcement learning for wake-up scheduling. In *Proceedings of the 7th european conference on wireless sensor networks.*

MirHassani, S. A. (2006). Improving paper spread in examination timetables using integer programming. *Applied Mathematics and Computation*, 179:702–706.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Morena Pérez, J. A., Marcos Moreno-Vega, J., and Rodríguez Martín, I. (2003). Variable neighbourhood tabu search and its application to the median cycle problem. *European Journal of Operational Research*, 151:365–378.

Moscato, P. and Cotta, C. (2003). *Handbook of meta-heuristics*, chapter A gentle introduction to memeticalgorithms, pages 105–144. Kluwer Academic Publishers.

Müller, T. (2008). Itc 2007: Solver description. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008.*

Müller, T. (2009). Itc2007 solver description: a hybrid approach. *Annals OR*, 172(1):429–446.

Müller, T., Barták, R., and Rudová, H. (2004). Conflict-based statistics. In Gottlieb, J., Landa Silva, D., Musliu, N., and Soubeiga, E., editors, *EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics.* University of Nottingham.

Nareyek, A. (2003). *Metaheuristics: Computer decision-making*, chapter Choosing search heuristics by non-stationary reinforcement learning., pages 523–544. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Obit, J. H., Landa-Silva, D., Ouelhadj, D., and Sevaux, M. (2009). Non-linear great deluge with learning mechanism for solving the course timetabling problem. In *Proceedings of the 8th metaheuristics international conference (MIC 2009), Hamburgh Germany.*

Özcan, E. and Alkan, A. (2007). A memetic algorithm for solving a timetabling problem: An incremental strategy. In *Proceeding of the 3rd Multidisciplinary International Conference On Scheduling: Theory and Applications (MISTA07), Paris, France.*

Özcan, E., Bykov, Y., Birben, M., and Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristic. In *Proceeding of the 2009 IEEE Congres on Evolutionary Computation (CEC'09)*, pages 997–1004.

Özcan, E. and Ersoy, E. (2005). Final exam scheduler - fes. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1356–1363.

Özcan, E., Misir, M., Ochoa, G., and Burke, E. K. (2010). A reinforcement learning: great deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing*, 1(1):39–59.

Pais, T. C. and Burke, E. (2010). Choquet integral for combining heuristic values for exam timetabling problem. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland*, pages 305–320.

Paquete, L. F. and Fortseca, C. M. (2001). A study of examination timetabling with multiobjective evolutionary algorithms. In *Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, pages 149–154.

Parkes, A. and Özcan, 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.

Petrovic, S. and Burke, E. (2004). University timetabling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chapter 45.* Chapman Hall/CRC Press.

Petrovic, S. and Bykov, Y. (2003). A multiobjective optimisation technique for exam timetabling based on trajectories. In Burke, E. K. and De Causmaecker, P., editors, *Lecture notes in computer science: Practice and theory of automated timetabling IV: selected papers from the 4th international conference*, volume 2740, pages 179–192. Berlin: Springer.

Pillay, N. (2008). A developmental approach to the examination timetabling problem. In *Proceedings of the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008.*

Pillay, N. (2009a). Evolving hyper-heuristics for the uncapacitated examination timetabling problem. In *Proceedings of the 4th Multidisciplinary International Con* *ference on Scheduling: Theory and Applications (MISTA'09), Dublin, Ireland*, pages 447–457.

Pillay, N. (2009b). The revised developmental approach to the uncapacitated examination timetabling problem. In *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 187–192.

Pillay, N. (2010a). Evolving hyper-heuristics for a highly constrained examination timetabling problem. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland*, pages 336–346.

Pillay, N. (2010b). Evolving hyper-heuristics for a highly constrained, multi-objective examination timetabling problem. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland*.

Pillay, N. and Banzhaf, W. (2007). A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In Neves, J., Santos, M., and Machado, J., editors, *Lecture Notes in Computer Science: Progress in Artificial Intelligence, 13th Portuguese Conference on Aritficial Intelligence, EPIA Workshops 2007*, volume 4874, pages 223–234. Springer.

Pillay, N. and Banzhaf, W. (2008). A developmental approach to the uncapacitated examination timetabling problem. In *Lecture notes in computer science: Parallel Problem Solving from Nature PPSN X: selected papers from the international conference*, volume 5199, pages 276–285.

Pillay, N. and Banzhaf, W. (2009). A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. *European Journal of Operational Research*, 197(2):482–491.

Pirlot, M. (1996). General local search methods. *European Journal of Operational Research*, 92:493–511.

Qu, R., Burke, E., 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. and Burke, E. K. (2005). Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems. In *Proceedings of the 6th Metaheuristic International Conference 2005*, Vienna, Austria.

Qu, R. and Burke, E. K. (2007). Adaptive decomposition and construction for examination timetabling problems. In *Proceedings of the Multidisciplinary international scheduling: theory and applications (MISTA'07), Paris, France*, pages 418–425.

Qu, R. and Burke, E. K. (2009). Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. *Journal of Operational Research Society*, 60:1273–1285.

Qu, R., Burke, E. K., Mccollum, B., Merlot, L. T., and Lee, S. Y. (2009b). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1):55–89.

Qu, R., He, F., and Burke, E. K. (2009c). Hybridizing integer programming models with an adaptive decomposition approach for exam timetabling problems. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009*, pages 435–446, Dublin, Ireland.

Ross, P. (2005). *Search methodologies: Introductory tutorials in optimisation and decision support techniques*, chapter Chapter 17: Hyper-heuristics, pages 529–556. Springer.

Ross, P. and Corne, D. (1995). Comparing genetic algorithm, simulated annealing and stochastic hill climbing of timetabling problems. In *Proceedings of the 1995 AISB Workshop on Evolutionary Computing, Lecture Notes in Computer science*, volume 993, pages 94–102. Springer-Verlag.

Ross, P., Corne, D., and Terashima-Marin, H. (1996). The phase transition niche for evolutionary algorithms in timetabling. In Burke, E. and Ross, P., editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. Lecture Notes in Computer Science*, volume 1153, pages 309–324.

Ross, P., Hart, E., and Corne, D. (1998). Some observations about ga-based exam timetabling. In Burke, E. and Carter, M., editors, *In Practice and Theory of Automated Timetabling: Selected Papers from the 2nd International Conference. Lecture Notes in Computer Science*, volume 1408, pages 115–129.

Ross, P., Marin-Blázquez, J., and Hart, E. (2004). Hyper-heuristics applied to class and exam timetabling problems. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, pages 1691–1698.

Sabar, N. R., Ayob, M., and Kendall, G. (2009). Tabu exponential monte-carlo with counter heuristic for examination timetabling. In *Proceedings of 2009 IEEE symposium on computational intelligence in scheduling (CISched 2009), 30 Mar - 2 Apr, 2009, Nashville, Tennessee, USA*, pages 90–94.

Sastry, K., Goldberg, D., and Kendall, G. (2006). *Introductory tutorials in optimisation, decision support and search methodology*, chapter 4: Genetic algorithms, pages 97–125. Springer.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127.

Schmidt, G. and Strohlein, T. (1980). Timetable construction-an annotated bibliography. *The Computer Journal*, 23:307–316.

Shaker, K. and Abdullah, S. (2009). Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems. In *Proceedings of the second Data Mining and Optimization, 2009. DMO '09*, pages 149 –153.

Sheibani, K. (2002). An evolutionary approach for the examination timetabling problems. In Burke, E. and Causmaecker, P. D., editors, *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. 21st-23rd August 2002. KaHo St.-Lieven, Gent, Belgium*, pages 387–396.

Sierksma, G. (2001). *Linear and integer programming: Theory and practice.* New York: Marcel Dekker, Inc., 2nd edition.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: an introduction.* The MIT Press, Cambridge.

Terashima-Marin, H., Ross, P., and Valenzuela-Rendon, M. (1999). Application of the hardness theory when solving the timetabling problem with gas. In *Proceedings of the Congress on Evolutionary Computation 1999, Washington, D.C*, pages 604–611.

Thompson, J. and Dowsland, K. (1996a). Variants of simulated annealing for the examination timetabling problem. *Annals of Operational Research*, 63:105–128.

Thompson, J. and Dowsland, K. A. (1996b). General cooling schedules for a siimulated annealing based timetabling system. In *Lecture notes in computer science: Practice and theory of automated timetabling I: selected papers from the 1st international conference*, pages 345–363, London, UK. Berlin-Springer.

Thompson, J. M. and Dowsland, K. A. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7-8):637–648.

Trick, M. A. (2011). Optimization and its applications. *Hybrid Optimization*, 45:489–508.

Tuga, M., Berrettan, R., and Mendes, A. (2007). A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In *ACIS-ICIS'07*, pages 400–405.

Turabieh, H. and Abdullah, S. (2011). An integrated hybrid approach to the examination timetabling problem. *Omega*, 39:598–607.

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. (2001). Examination timetables and tabu search with longer-term memory. In E. K., B. and Erben, W., editors, *Lecture notes in computer science. Practice and theory of automated timetabling III: selected papers from the 3rd international conference*, volume 2079, pages 85–103. Berlin: Springer.

White, G. M., Xie, B. S., and Zonjic, S. (2004). Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, 153 (1):80–91.

Wong, T., Bigras, P., and de Kelper, B. (2005). A multi-neighborhood and multi-operator strategy for the uncapacitated exam proximity problem. In *Proceeding of IEEE International conference on Systems, Man and Cybernatic*, volume 4, pages 3810–3816.

Wong, T., Côté, P., and Gely, P. (2002). Final exam timetabling: A practical approach. In *Proceedings of the IEEE Canadian conference on electrical and computer engineering (CCECE 2002)*, volume 2, pages 726–731.

Wood, D. C. (1968). A system for computing university examination timetables. *The Computer Journal*, 11(1):41–47.

Wood, D. C. (1969). A technique for colouring a graph applicable to large scale timetabling problems. *The Computer Journal*, 12(4):317–319.

Wren, A. (1996). Scheduling, timetabling and rostering - a special relationship? In Burke, E. K. and Ross, P., editors, *Lecture Notes in Computer Science:The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I)*, volume 1153, pages 46–75. Springer-Verlag.

Yang, Y. and Petrovic, S. (2004). A novel similarity measure for heuristic selection in examination timetabling. In *Proccedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2004)*, Pittsburg.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8:338–353.

Zeleny, M. (1974). A concept of compromise solutions and method of displaced ideal. *Computers & Operations Researh*, 1(4):479–496.