Novel Hyper-heuristic Approaches in Exam Timetabling

Amr Soghier, MSc.

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

July 2012

Abstract

This thesis presents an investigation into the use of hyper-heuristic approaches to construct and improve solutions for exam timetabling problems. The majority of the approaches developed in the literature operate on a space of potential solutions to a problem. However, a hyper-heuristic is a heuristic which acts on a space of heuristics, rather than a solution space directly. The majority of research papers, so far, have involved the design of tailored approaches which focus on tackling specific problems. Less well studied are hyper-heuristics which can operate on a range of problems.

The first area of investigation is the use of a Greedy Random Adaptive Search Procedure to combine multiple heuristics within the construction of timetables. Different combinations of multiple heuristic orderings were examined, considering five graph-based heuristic orderings - Largest Degree, Saturation Degree, Largest Enrolment, Largest Coloured Degree and Largest Weighted Degree. The development utilised two heuristic orderings simultaneously and subsequent development went on to break ties within the heuristic orderings.

Improving exam timetables has been the subject of much research. It is generally the case that a meta-heuristic is used to improve constructed solutions. Indeed, there are usually a very large number of different methods

to obtain better timetables. Some method is required to automatically choose the appropriate method to be applied depending on the constraints in a problem. In response to that demand, the second area of investigation of this thesis is concerned with a hyper-heuristic approach to hybridise neighbourhood operators to improve constructed timetables.

The reuse of heuristics in different problem domains have been explored previously in the literature. Bin packing heuristics on their own are simple techniques where items in the problem are packed using a specific strategy to construct solutions. Therefore, using bin packing heuristics to assign exams to time slots and rooms have been investigated. This is the first time that bin packing heuristics are used within a hyper-heuristic to assign exams to rooms and time slots.

Acknowledgements

My sincere thanks go to my two supervisors, Dr Rong Qu, and Professor Edmund Burke, for the encouragement and flexibility they have given to me over the last few years. Through their support and trust, I have had the opportunity to work on an interesting project and to learn alot from their advice and guidance.

Special thanks to my mom, my sisters Lamia and Israa, and my aunts for their love, patience and support throughout the study.

I would also like to thank the EPSRC, whose funding made this project possible. In addition, many thanks go to the rest of the academic and administrative staff at the Automated Scheduling, optimisation and Planning (ASAP) research group.

Finally, I would like to thank Sally Hankin, Professor Chris Hawkey from the Queens Medical Centre of Nottingham for their support during my illness and for their help to overcome stress during my study.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background and Motivation

For more than 40 years exam timetabling has become one of the mostly studied domains in the AI and OR research communities. This is due to its importance in many academic institutions worldwide. The basic problem is to allocate a time slot and a room for all the exams within a limited number of permitted time slots and rooms in order to find a feasible timetable. This assignment process is subject to 'hard' constraints which must be satisfied in order to get a *good* timetable. An example of such constraint is that no student is required to attend two exams at the same time. On the other hand, other constraints can be violated but must be satisfied as much as is possible to obtain a good timetable. These are called 'soft' constraints. Not using certain rooms is an example of a soft constraint.

As this task is time consuming and tedious to carry out manually, much effort during the last few decades has been directed to generate timetables automatically. With a large number of events needing to be assigned to

resources (time slots and rooms) and a list of constraints (both hard and soft) needing to be addressed, there are a large number of potential solutions to this problem. Therefore, much of the research has been aimed at developing methodologies that focus on producing the best quality timetables for a specific problem or even a specific problem instance. A more recent direction in this field, namely, hyper-heuristics, aims to raise the level of generality of search methodologies to create algorithms that act well over a range of problems. A hyper-heuristic is seen as a heuristic to choose heuristics.

The motivation for the work presented in this thesis is to investigate whether the use of hyper-heuristics could be of benefit in automating the decision making process in the construction and improvement of solutions to the exam timetabling problem.

1.2 Aims and Scope

The first area of investigation, described in chapter 4, is an exploration of how a Greedy Random Adaptive Search Procedure (GRASP) can be employed in a hyper-heuristic to combine multiple heuristics to construct exam timetables. GRASP is a meta-heuristic approach which is used to solve hard combinatorial optimisation problems. It is a two phase procedure which starts by adaptively creating a randomised greedy solution. As this solution produced is not guaranteed to be locally optimal, an improvement phase is required. Therefore after the solution is constructed, an iterative improvement is applied until a locally optimal solution is found [85, 86].

During the process of construction, the order in which exams are assigned to time slots has been shown to have a major effect on the construction process. An assessment of how difficult it is to place a given exam into a timetable is used to guide the order of placement. The usual strategy is to place the most difficult exams first, on the basis that it is better to leave the easier exams until later in the process when there are fewer time slots remaining. There are many different criteria that may be used when assessing this difficulty. A common approach has been to employ graph based heuristics.

Several heuristics were developed to solve graph colouring problems [20]. The idea of these heuristics is based on estimating difficulty and colouring the most difficult vertices first in a simple constructive technique. Exam timetabling problems with only hard constraints can be represented as graph colouring models. Vertices in the graph represent exams in the problem, and edges represent the conflicts between examinations. Colours correspond to time slots. This measure is then used to determine the order in which the exams are assigned into the timetable and, hence, are referred to as 'heuristic orderings'. Examples of such heuristics are the number of available time slots to schedule an exam, the number of students enroled on each exam, etc. Detailed descriptions of these heuristic orderings are given in section 2.2.2.

In this thesis, for the first time, a GRASP is used within a hyper-heuristic to combine multiple heuristics simultaneously in order to provide a measure of the difficulty of placing each exam. This measure is then used to order the exams for assignment. Various combinations of heuristics are investigated in the construction process.

Graph heuristics have been hybridised in an iterative adaptive approach

developed in [29]. In each iteration, the exams are ordered using different graph heuristics and the exam to be scheduled is chosen. In addition, the exams that are found to be difficult to schedule in previous iterations are adaptively moved forward. Another approach has been developed in [142] where different graph heuristics are adaptively hybridised. Different ordering strategies in graph heuristics are adaptively called at a higher level. However, a random exam is selected in the situation where a tie appears. Hence, instead of choosing these exams randomly, tie breaking is a way to help guide a search in scenarios where two or more exams have the same weight in a specific ordering. Therefore, the use of tie breakers in heuristic orderings is investigated in chapter 5. In addition, an exam timetabling instance generator was developed as part of this thesis to test the adaptiveness of the tie breaking on randomly generated data.

The third area of investigation, described in chapter 6, is combining heuristic moves to improve the quality of constructed timetables. It is generally the case that a number of alternative solutions that satisfy all the hard criteria are possible. Indeed, there is usually a very large number of such feasible solutions. Therefore, some method is required to automatically improve the overall quality of the different solutions and select the best.

Finally, in exam timetabling problems which consist of time slot assignment and room allocation, the choice of time slots and rooms has a great effect on the quality of the solution generated according to the soft constraints in a problem. Therefore, a means of combining heuristics and evaluating the performance when assigning an exam to a certain time slot and room is required. In chapter 7, an investigation on combining bin packing heuristics and their effect on the quality of the solutions is presented.

There are a number of objectives that were addressed in order to accomplish

the primary aim of the research which can be outlined as follows:

1. to investigate the ability of hyper-heuristics to generalise over a set of exam timetabling problems.
2. to explore the use of GRASP within a hyper-heuristic to combine heuristics simultaneously to construct timetables.
3. to investigate the effectiveness of applying tie breakers to heuristic orderings within a hyper-heuristic.
4. to explore the hybridisation of heuristic moves within a hyper-heuristic to improve timetables.
5. to investigate the performance of a hyper-heuristic which combines bin packing heuristics to assign exams to time slots and rooms.

1.3 Structure of the Thesis

The thesis is structured as follows. Chapter 2 provides a description of the exam timetabling problem and describes the benchmark datasets that are used in this research. A review of the different algorithms and approaches investigated on exam timetabling problems with focus on the approaches applied to the benchmarks is also presented. Furthermore, a description of the objective functions currently used to evaluate the quality of the timetables is described.

Chapter 3 presents a review of the relevant literature for hyper-heuristics. Chapter 4 presents a description of a hyper-heuristic approach which constructs exam timetables. Chapter 5 presents a study on the effectiveness of breaking ties in heuristic orderings. Chapter 6 presents a hyper-heuristic approach to improve timetables using heuristic moves. Chapter 7 presents the most recent work, a hyper-heuristic system which focuses on time slot and room assignment using both graph colouring and bin packing heuristics. Finally, chapter 8 provides some concluding remarks and suggestions for future research that arise from the work presented in this thesis.

1.4 List of publications

The following academic papers have been produced as a result of this research. For each paper, the corresponding chapter in this thesis which describes the content of the publication is presented.

- Chapter 4

  E.K. Burke, R. Qu, and A. Soghier. 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 93-104, 2009.[48]

- Chapter 5

  E.K. Burke, R. Qu, and A. Soghier. Adaptive tie breaking and hybridisation in a graph-based hyper-heuristic for exam timetabling problems. In *Proceedings of the Nature Inspired Cooperative Strategies for Optimisation*, Studies in Computational Intelligence. Springer, 2011.[50]

- Chapter 6

  E.K. Burke, R. Qu and A. Soghier. Adaptive Selection of Heuristics for Improving Constructed Exam Timetables. *The 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010)*, 10-13 August 2010, Belfast, Northern Ireland.[49]
  An extended version of this paper has been submitted to Annals of OR (ANOR)

- Chapter 7

  E.K. Burke, R. Qu and A. Soghier. Adaptive Selection of Heuristics for Assigning Time Slots and Rooms in Exam Timetables. Paper submitted to *The Journal of Applied Intelligence.*

Chapter 2

Exam Timetabling

2.1 Introduction

Exam timetabling [54, 144] is the process of assigning a number of exams into a limited number of time slots putting in consideration a number of constraints that must be satisfied. Many academic institutions face problems in scheduling their exams every semester or term. The greater the number of constraints in the problem, the more difficult the exam-timetabling problem becomes. In addition, there is a great number of different constraints that make exam-timetabling problems different from one institution to the other. It is vital that some constraints are fully satisfied. These are called hard constraints. For example, a student cannot attend two exams at the same time. Therefore, any two exams that are attended by the same student must not be scheduled at the same time. A timetable that satisfies all hard constraints is called a feasible timetable.

Other constraints could be violated but the higher the degree of satisfying such constraints the better the solution. These are called soft constraints.

For example, providing students with more gaps between their exams is very desirable. However, not providing the gaps will not affect the exam being held and the timetable followed. Violations of soft constraints could be used to decide the quality of the timetable created.

Every institution will have its own set of constraints according to its preferences. An institution could decide that each student should have only one exam per day, which is regarded as a hard constraint. Another institution would accept that students can have more than one exam per day. In this case, the constraint of having a single exam each day is no longer a hard constraint.

Examples of common hard constraints are:

- avoid any student having an exam scheduled in the same time slot;
- all exams must be assigned to a time slot and a room during the exam period;
- the capacity of a room should not be exceeded at a given period.

In practice, the quality of a feasible timetable is evaluated in a different way. In the majority of the cases, the measure of quality is calculated based upon a penalty function which represents the degree to which the soft constraints are violated. Examples of soft constraints are as follows:

- exam X must be after exam Y or exam A must use room R;
- a student must not sit two exams on the same day;
- large exams should not be scheduled close to the end of the exam session;

- certain rooms or time slots should not be used;
- exams with different durations should not be scheduled in the same room.

More details on constraints for exam timetabling can be found in [23, 74, 124, 144].

Many techniques over the years have been based on graph colouring [31, 54, 71]. This is the process of assigning colours to the vertices in a graph so that no adjacent vertices share the same colour. The objective is to use the minimum number of colours possible. This minimum number of colours is known as the chromatic number of a graph. Exam timetabling problems with only hard constraints can be represented as graph colouring models. Vertices in the graph represent exams in the problem, and edges represent the conflicts between exams. Colours correspond to time slots. Hence, if the exam timetabling problem is considered as a graph colouring problem, the aim is to find the minimum number of time slots which are able to accommodate all the exams without any clashes. Several heuristics (e.g. [20]) have been developed to solve graph colouring problems. Graph colouring heuristics are still widely used to develop new methods to solve timetabling problems [31].

By analysing the student enrolment list, the exams that are in conflict(exams that have at least one common student) can be identified. Cole[58] represented the conflicting exams using the incompatibility table, while Broder[21] used the term conflict matrix to define the same thing. The conflicting exams are represented by a conflict matrix, $C = [c_{ij}]_{NxN}$ where $i, j \in 1, ..., N$ (N is the number of exams). Element c_{ij} denotes the number of students enroled for both exam i and exam j. When a nonweighted graph is em-

ployed, it is also possible to use $c_{ij} = 1$ if there is conflict between exam i and exam j; $c_{ij} = 0$ otherwise. It is a symmetrical matrix, i.e. element $c_{ij} = c_{ji}$. For diagonal cells (i.e. $i = j$), each cell either contains the number of students enroled for the particular exam (c_ij = number of students for exam i) or the cell contains zero ($c_{ij} = 0$) to denote that there is no conflict. Essentially, several pieces of information can be generated from the conflict matrix that are related to graph theory. The number of exams in conflict for an exam is equivalent to the node degree. Node degree values are utilised when heuristic orderings (e.g. Largest Degree, Largest Coloured Degree and Largest Weighted Degree, see section 2.2.2) are employed to order the exams by difficulty when constructing solutions. It is also possible to use the diagonal cell values for the heuristic ordering Largest Enrolment if c_{ij} is not set to zero when $i = j$.

Constraint based techniques have attracted the attention of researchers due to their ease and flexibility when employed to exam timetabling problems. Exams are modeled as variables with finite domains. The values within the domains represent the time slots and rooms. The values are assigned in a sequential manner to construct solutions for the problems. David [69] applied constraint satisfaction to model an exam timetabling problem in a French school, the Ecole des Mines de Nantes. Partial solutions were constructed first since time complexity was crucial. Local repair strategies were employed successively to obtain complete solutions and make improvements. The approach was run several times using different initial assignments to lower the chance of missing good solutions.

Duong and Lam [82] also employed constraint programming to generate initial solutions for the exam timetabling problems at the HoChiMinh City University of Technology. The generated solutions were improved using

a Simulated Annealing approach (see section 2.2.2). Backtracking and forward checking were used to enhance the search. A labeling strategy, which indicates the order in which variables are to be initiated, was used to order the variables (exams) by a number of factors such as the number of students and the size of the domain.

Bullnheimer [22] used Simulated Annealing on two neighbourhood structures on small scale exam timetabling problems. He adapted a model for Quadratic Assignment Problems to formulate a problem at the University of Magdeburg. The models enabled the university administrators to control the spacing between conflicting exams.

Sheibani [150] developed a mathematical model which is used by a Genetic Algorithm (see section 2.2.2) to solve exam timetabling problems in training centres with the objective of maximising the gaps between exams. The closeness between exams was estimated using an activity-on-arrow network which was used in the fitness function of the Genetic Algorithm.

Wong et al. [167] applied a Genetic Algorithm to a solve an exam timetabling problem at the Ecole de Technologie Superieure. The problem was modeled as a Constraint Satisfaction problem. Parents were selected using tournament selection and repairing strategies were combined with mutations to produce better candidates.

Colijn and Layfield [59] applied a multi-stage approach for the exam timetabling problem in the University of Calgary. In the 1^{st} stage, exams were moved to reduce the number of occurrences where students sit two exams in a row. In the 2^{nd} stage, the same approach is used to reduce the number of occurrences of students taking three and four exams in a row.

Burke et al. [30] used another multi-stage approach to deal with nine

criteria in exam timetabling problems (e.g. room capacity, time and order of exams, proximity of exams, etc). In the 1^{st} stage, the criteria was dealt with individually using Saturation Degree (see section 2.2.2) to generate a set of feasible solutions. The initial solutions were improved simultaneously using heuristics.

This chapter will provide a review of the exam timetabling literature which is relevant to the work studied in this thesis. There are three main exam timetabling problems which are used to evaluate the performance of the investigated approaches. The exam timetabling problems are the Toronto benchmark and the Exam Timetabling Competition (ITC2007) dataset. In this chapter we review the previous work on exam timetabling problems by organising the subjects into two sections. Section 2.2 presents the Toronto benchmark literature and a brief description of the different approaches developed to solve exam timetabling problems. However, the section focuses on the approaches applied to the Toronto benchmark. Finally, section 2.3 presents the ITC2007 exam timetabling problem and the approaches applied to the dataset. The aim of this chapter is to put this contribution in context, and provide information about the problems investigated in this thesis and the different approaches applied in the area of research. For further information a survey for exam timetabling can be found in [144].

2.2 The Toronto Benchmark

2.2.1 Problem Description

In 1996, Carter et al. introduced a set of exam timetabling benchmark data. This dataset is widely used in exam timetabling research by different

state-of-the-art approaches and can therefore be considered as a benchmark [54]. It consists of 13 real-world problems from 3 Canadian high schools, 5 Canadian, 1 American, 1 UK and 1 mid-east universities. The problem has one hard constraint where conflicting exams cannot be assigned to the same time slot. In addition, a soft constraint is present to minimise the number of exams assigned in adjacent time slots in such a way as to reduce the number of students sitting exams in close proximity. The sum of proximity costs is given as follows:

$$\sum_{i=0}^{4}(w_i \times L)/S$$

where

- $w_i = 2^{4-i}$ is the cost of assigning two exams with i time slots apart. Only exams with common students and are four or less time slots apart cause violations
- L is the number of students involved in the conflict
- S is the total number of students in the problem

There has, however, been a problem with these benchmarks. Over the years two slightly different versions of some of the sets have been made available. This has caused some confusion when comparing different approaches in the literature. Qu et al. [144] have recently completed a thorough re-classification of all problem instances which have appeared in published work. In the version II problem instances, 3 instances (sta83 II, yor83 II and ear83 II) contain repetitions in the data files. Therefore, these repetitions were removed [144] and named as version IIc (the "c" stands for "corrected"). A modification to the number of time slots had to be

made for sta83 II from 13 to 35 time slots to obtain a feasible solution. The characteristics of the versions used in this research are presented in table 2.1. The original version I, version II and the version IIc data files are all available at http://www.asap.cs.nott.ac.uk/resources/data.shtml.

For each data instance, two files are provided - a student data file (with a .stu file extension) and exam data file (with a .exm file extension). A list of the exams enrolled on by each student are stored in the student data file, while the number of available time slots is stored in the exam data file.

Examples of the information available are the conflicting exams, the total number of students enrolled for each exam and the number of examinations enrolled by each student. This information can be used to measure the density of conflicting exams for each problem instance. The conflict density values shown in the table indicates the density of conflicting exams. To calculate the conflict density, a conflict matrix C is defined in which each element c_{ij} is one if exam i conflicts with exam j (at least one student is enrolled for both exam i and j), or zero otherwise. The conflict density is calculated by summing the number of other exams that each exam is conflicted with (i.e. the elements of the conflict matrix for which $c_{ij} = 1$), and dividing the sum by the total number of elements in the conflict matrix.

Table 2.1: Characteristics of the Toronto benchmark dataset

Problem	Exams I/II/IIc	Students I/II/IIc	Enrolments I/II/IIc	Density	time slots
car91 I	682	16925	56877	0.13	35
car92 I	543	18419	55522	0.14	32
ear83 I	190	1125	8109	0.27	24
ear83 IIc	189	1108	8057	0.27	24
hec92 I	81	2823	10632	0.42	18
hec92 II	80	2823	10625	0.42	18
kfu93 I	461	5349	25113	0.06	20
lse91 I	381	2726	10918	0.06	18
sta83 I	139	611	5751	0.14	13
sta83 IIc	138	549	5417	0.19	35
tre92 I	261	4360	14901	0.18	23
uta92 I	622	21266	58979	0.13	35
uta92 II	638	21329	59144	0.12	35
ute92 I	184	2750	11793	0.08	10
yor83 I	181	941	6034	0.29	21
yor83 IIc	180	919	6002	0.3	21

2.2.2 Approaches for the Toronto benchmark

Approaches developed to solve timetabling problems usually consist of two phases [107]. In the first phase, a solution is constructed by using a sequential construction algorithm. The constructed solution can be feasible or infeasible. If a solution is infeasible, it can be corrected during the second phase which is an iterative improvement phase.

In the second phase, the initial solution is improved while ensuring the feasibility of the solution. The improvements can be implemented by using any search algorithm such as Genetic Algorithms [101], Tabu Search [95–97], Simulated Annealing [117] or the Great Deluge Algorithm [81]. In this section, a brief description of various search algorithms and approaches applied to exam timetabling problems is presented.

In the first two parts of this research, the focus is on the construction process, as constructing feasible solutions is a difficult task especially for large, real-world timetabling problems [107]. A detailed explanation of the construction algorithms employed in this research are outlined in the following section.

Sequential Constructive Algorithms

Sequential constructive algorithms were amongst the first approaches used to tackle the exam timetabling problem in an automated manner [21, 58]. The idea behind such algorithms is to prioritise the assignment of the exams that might cause problems if left to be scheduled later in the process. Therefore, the overall aim is to avoid the assignment of exams to time slots which might later lead to an infeasible solution. An infeasible solution is

obtained when at least one exam remains unscheduled. In many cases this is because all the potentially valid time slots are invalidated. In such a situation, a different ordering may enable a feasible solution to be reached.

Approaches which order the exams prior to assignment to a period have been discussed by several authors including Boizumault [18], Brailsford [19], Burke and Newall [43], Burke et al. [31], Burke and Petrovic [46], and Carter et al. [54]. The strategy of solving exam timetabling problems sequentially has been applied to the Toronto benchmark by Carter et al. [54], Burke and Newall [43], and Burke et al. [42]. Usually, the unscheduled examinations are ordered according to certain criteria which represent the priority to schedule exams (the most difficult first). A number of well known strategies from the graph colouring problem were used. Many studies employ graph theory to calculate the "difficulty to schedule an exam". The following list describes the most common graph colouring based heuristic orderings used in timetabling research:

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

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

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

Largest Coloured Degree (LCD) First. This heuristic is based on LD. For this heuristic, only exams which have been already assigned to the

schedule are considered as the exams which will cause conflict.

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

Largest Uncoloured Degree(LUD) First. Exams are ranked in an increasing order by the number of conflicts each exam has with examinations unscheduled yet.

Largest Uncoloured Weighted Degree(LUWD) First. This heuristic is based on LUD. Besides the number of exams in conflict, the total number of students involved in the conflict is taken into account as well.

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

In 1961, Appleby et al. [5] implemented graph colouring techniques in the preparation of school timetables. Graph based heuristic orderings were also applied to other educational timetabling problems. LD was the most widely used heuristic ordering in earlier research into exam timetabling

[21, 58, 161]. Furthermore, the LE and LD heuristic orderings were used in [168]. In this approach, the exams which require the room with the largest capacity were selected. These exams were then ordered using LD and the first exam in the ordering is scheduled. The same process was repeated using the second largest room and so on. LE and LD were also combined simultaneously through a simple linear combination multiplied by a weighted factor w_{LE} that was varied [111].

In 1979, Saturation Degree was introduced to solve the graph colouring problem [20]. Saturation degree suggests that the most difficult vertex to colour is the vertex with the least available colours. Mehta [125] used the SD heuristic to schedule exams in twelve time slots. However, to avoid any student from sitting two exams at the same time, thirteen time slots were required. In addition, an exam grouping mechanism and an adjustment to the sequence of time slots was applied to avoid conflicts and to spread out each student's schedule.

LD, SD, LWD and LE were used in [122] and [54] to rank the exams in a decreasing/increasing order to estimate the difficulty of scheduling each of the exams. An ordering was obtained using each one of the four heuristics. Then, the exams were selected sequentially and assigned to a time slot without violating any hard constraint. In [54], the algorithm presented was used to find the maximum clique of conflicting examinations. A clique of exams is a group of exams in which each exam conflicts with every other exam. The size of the maximum clique is very important as it can be used to determine the minimum number of time slots required to schedule all the exams [93]. The approach starts by scheduling the exams in the maximum clique in different time slots. Then the rest of the exams were scheduled using the heuristic orderings. This approach was applied to ten

random instances and thirteen real problems. The use of cliques for exam timetabling has been further investigated in [53].

Greedy Randomised Adaptive Search Procedure (GRASP) is a meta-heuristic which has attracted significant recent attention [86]. It is a two phase procedure which starts by adaptively creating a randomised greedy solution followed by an improvement phase. A study on the efficiency of the four heuristic orderings (i.e. Largest Degree, Saturation Degree, Largest Weighted Degree and Largest Enrolment) was presented in [55]. The orderings were used to construct the initial solutions in the first phase of a GRASP algorithm. A roulette wheel selection was then applied to the top n exams to choose the next exam to be scheduled. The value of n was chosen depending on the number of exams in the problem instance. The exam chosen was scheduled into the first time slot which satisfied all the hard constraints.

A priority formula was used to order the exams in [89]. The priority formula uses information on the exams extracted from the problem. In addition, a manual special priority setting was employed to override other soft constraints. For example, final year exams were given special priority and were scheduled first. The aim of scheduling exams in a sequential manner using certain criteria or heuristics is to make sure the timetabling process ends by scheduling all the exams in the construction phase. However, in some cases the exams are not assigned in the first attempt. Therefore, different strategies were used to select which time slot to choose when assigning an exam. This can have a significant effect on the construction process. Some common strategies mentioned in the literature are as follows:

- Using the first or last valid time slot.

- Using a random valid time slot.
- Using the time slot leading to the least penalty cost.
- Using the time slot leading to the least number of unused seats.

After applying different strategies of event selection and time slot assignment, various approaches can be applied in the case where a feasible timetable is not achievable. The idea is to reshuffle the earlier scheduled events to find a feasible solution. Backtracking was implemented in [54] and [122] whenever a valid time slot could not be found to schedule an event. In order to free a time slot, the time slot with the minimum number of conflicting scheduled exams was chosen. The minimum disruption cost (the cost of reshuffling the conflicting examinations from the selected time slot into another valid time slot and inserting the current unscheduled exam into the selected time slot) is calculated to identify which exam was to be moved. All the conflicting exams were either moved to a different valid time slot with the least penalty cost or returned to the list of unscheduled exams. To avoid an infinite loop, an exam is not allowed to be returned to the list of unscheduled exams more than three times. This process ends when a feasible solution was produced. A similar backtracking approach was investigated in [55].

Another approach proposed in [43] applied an adaptive ordering strategy in which the exams were dynamically ordered by a particular heuristic. This heuristic could then be modified to take into account the penalty caused from constraint violations. Their work was motivated by the Squeaky Wheel approach introduced by Joslin and Clements [112]. Other approaches in the literature implemented heuristic orderings where events were split into independent sets. The events are split in a way to make sure there are

no conflicts within a group. Each event in the group is then assigned a time slot with the objective of minimising the violations of soft constraints. One of the earliest approaches using this technique was introduced in [169]. A comparison between two different groupings for the graph colouring problem was presented. The first was based on the graph heuristic LD, while the second was based on a similarity matrix. A similarity matrix was generated based on the information obtained from the conflict matrix. As defined in [169], "if vertices i and j are not connected, the similarity is the number of other vertices k which are connected to both i and j". The experiments on timetabling problems showed that the ordering using the LD heuristic performed better than the similarity matrix approach when it was applied to randomly generated datasets.

An automated exam timetabling system called HOREX employed by the L'ecole Polytechnique de Montrèal was presented in [73]. The system was implemented using five heuristic orderings to select exams and place them into non conflicted groups. The heuristic orderings included LE, two random approaches, and two orderings based on LD. Another approach was developed in [34] using LD to determine which examinations could be grouped together in the same time slot. The exams in each group were ordered using LE to assign them to rooms with the aim of minimising the number of unused seats. In this problem, more than one exam could be scheduled in the same room.

In summary, many different heuristic orderings were examined in the literature. It was proven that it is difficult to determine which ordering would be most suitable for a given problem [54]. Furthermore, the investigation presented in [43] suggested that adaptively changing the heuristic ordering during construction can produce better solutions. A common observa-

tion in different approaches implies that it might be beneficial to hybridise heuristics during construction in order to further improve the quality of the solutions obtained.

Iterative Improvement Methods

As stated earlier, after constructing a solution in the first phase, the solution is often improved in the second phase. The improvement is an iterative process where the main objective is to improve the quality of the solution at the end of the process. Meta-heuristic approaches have been the most commonly used in the literature for this improvement. An excellent general review of meta-heuristic approaches in combinatorial optimisation can be found in [17]. The aim of heuristic search techniques is to provide an efficient way of iteratively exploring the search space of a given problem. However, most methods will get stuck into local optima. A meta-heuristic is a technique to guide other heuristics to produce better solutions through leading them out of local optimum. The term meta-heuristics was first used by Fred Glover and he defines it as [98]:

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

While Osman and Kelly [131] defined it as: "A meta-heuristic is an iterative generation process which guides a subordinate heuristic ..."

This section focuses on the meta-heuristic approaches developed in the

literature for exam timetabling.

Simulated Annealing and the Great Deluge Algorithm

The process of annealing originally comes from physics where a material is heated then cooled gradually to allow atoms to arrange themselves in a configuration with lower internal energy. It was introduced in the early 1980s by [117] and [56].

Using a hill climbing algorithm to solve combinatorial optimisation problems could easily lead to falling into local optima. The reason this happens is that a hill climbing algorithm only accepts better solutions within a certain neighbourhood, without exploring other neighbourhoods, which may lead to a better solution. Simulated Annealing allows escaping from local optima by accepting worse solutions probabilistically.

In each iteration, Simulated Annealing replaces the current solution with a nearby random solution. If the nearby solution is better, then the solution is chosen. Otherwise the solution is chosen according to the difference between the costs of the last previous solutions, and the temperature parameter (T) that gradually decreases in each iteration. For minimisation problems, the Simulated Annealing process starts by randomly searching different neighbourhoods allowing the escape from local optima followed by a gradual decrease in the movement inside the search space.

Simulated Annealing has been applied to the exam timetabling problem to improve soft constraint satisfaction by Thompson and Dowsland [158, 159]. They focused on automating the tuning of the cooling schedule and adapting it to problem instances and their corresponding objective functions.

Burke et al. [27] investigated a variant of Simulated Annealing which is known as Great Deluge algorithm [81]. In comparison with the Simulated Annealing, the Great Deluge Algorithm accepts worse moves as long as the decrease in quality is below a certain level. It uses two parameters which are the decay factor and an estimate of the quality of the desired solution. The advantage of the Great Deluge Algorithm is that it requires less parameter tuning and therefore it can be used by people who are non-experts in meta-heuristics. The application of the Great Deluge Algorithm in exam timetabling problems has been investigated by Burke et al. [31] and, Burke and Bykov [23]. A comparison of Great Deluge Algorithm and Simulated Annealing applied to the Toronto benchmark instances, as reported by Burke et al. [27], and Abdullah and Burke [2], showed that, overall, Great Deluge Algorithm was superior to Simulated Annealing.

Tabu Search

Tabu search performs a local search process iteratively until a certain stopping condition is satisfied. Tabu search makes it possible to explore areas of the search space that have not been explored through keeping a record of the areas visited or preventing the reversal of certain moves. It also accepts worse moves to escape local optima. A tabu list is used for the storage of such information. The tabu list could be used in different ways to assist the local search, such as storing the moves that would lead to an undesirable solution or areas of the search space that have been previously explored. Tabu search has proved itself effective and efficient for solving combinatorial optimisation problems where it avoids stopping at suboptimal points and the repetition of previous solutions (see [94–97]).

Tabu Search has also been successfully applied to exam timetabling prob-

lems. Di Gaspero and Schaerf [74] investigated a family of Tabu Search algorithms and applied the algorithms to the exam timetabling benchmark datasets. The neighbourhoods used concerned those of which contributed to the violations of hard or soft constraints. The experimental results showed that the adaptive cost function and the effective selection of neighbourhoods concerning the violations were key features of the approach.

White and Xie [163] developed a four-stage Tabu Search algorithm which they called OTTABU. Both types of adaptive memory (namely recency-based short term memory and frequency-based longer term memory) were employed in order to construct exam timetables for the University of Ottawa. The approach focused on avoiding cycling with the aim of improving the quality of solutions. It was found that using long term memory can significantly improve Tabu search and produce better timetables. Later, the approach was extended in [162] and applied to twelve instances of the Toronto benchmark (see table 2.1). The comparison with the results published by other researchers showed that the performance of their approach was favourable.

Paquette and Stutzle [134] developed a Tabu Search methodology depending on giving priorities for constraints. They investigated two different ways of considering the constraints. In the first, the constraints were considered one at a time starting with the highest priority and breaking ties by considering the lower priorities. The second way considered all the constraints at a time starting from the highest priority. The second way obtained better results than the first. However, the first way proved to be more consistent. The length of the tabu list was determined according to the number of violations in the solutions. It was observed that the length of the tabu list needed to be increased with the size of the problems.

Evolutionary Algorithms

Evolutionary Algorithms (EA) are population based algorithms motivated by the process of natural evolution [109]. EAs depend on storing a number of solutions and creating better solutions by combining or changing the solutions in the current population. Amongst the popular EAs are Genetic Algorithms, Memetic Algorithms and Ant Algorithms.

Genetic Algorithms

Genetic Algorithms (GA) are designed through representing solutions of a problem as chromosomes, which later evolve to better solutions [101]. The chromosomes are encoded in a binary form as a series of 1s and 0s. The process starts by generating a random solution and evolving more generations of different solutions. The process of evolution happens in three steps which are selection, recombination and mutation. Individuals can be selected using fitness proportionate selection or ordinal selection. Fitness proportionate selection orders individuals according to their fitness then the individuals are randomly selected. Individuals with higher fitness have a higher chance of getting selected. Ordinal selection includes tournament and truncation selection. In a tournament, a certain number of chromosomes are chosen and are entered in a tournament against each other. The fittest is then chosen to become the parent. In truncation selection, the top (1/s)th individuals get s copies each in the mating pool.

After the individuals are selected and placed in the mating pool, they are recombined to form new chromosomes which are expectantly better. There are many methods to perform recombination such as k-point, uniform, order-based, uniform order-based, partially matched, cycle crossover.

Performing crossovers is usually not enough to explore the entire search space. Therefore, performing a mutation would add to the diversity of the chromosomes created. The simplest way to create a mutation is performing a bit flip. Finally, once a new generation is created, the current generation needs to be replaced. The current generation could simply be deleted and replaced with the same number of chromosomes just created. A steady-state replacement deletes a certain number of chromosomes from the current generation and replaces them. This could also be done without duplicating chromosomes that have been added.

Burke et al. [33] used GAs to minimise the number of time slots required for exam timetabling problems. They investigated using different selection heuristics and uniform crossover operators. Two of the heuristics were graph colouring heuristics (LD and LCD), one was a random, and the remaining were specially designed heuristics that highlighted the two constraints that needed to be addressed (i.e. the number of time slots and the spread of the conflicting exams) either individually or combined. These heuristic crossover operators were developed with the aim of avoiding infeasible timetables being produced during the recombination process. The experimental results showed that good quality timetables might be produced by integrating heuristics in crossover operators. Similar heuristic crossover operators were successfully implemented by Burke et al. [35] for another set of more difficult timetabling problems.

Ross et al. [147] discussed the weakness of direct coding in GAs. They suggested that a GA is more suitable for finding a good algorithm instead of directly searching for the solutions for a particular problem. Erben [84] applied grouping GAs for exam timetabling problems. In grouping GAs, a group of non-conflicting exams are treated as one gene. The chromosome

length represents the number of time slots for exam timetabling problems. In order to generate feasible solutions, the hard and soft constraints need to be incorporated in the crossover operator and mutation operator. Although the approach did not compete with the best results when applied to the Toronto benchmark, the computational time was less when compared to the other approaches. The work also highlighted the importance of representation when designing Genetic Algorithms.

Memetic Algorithms

The term '*Memetic Algorithm*' was introduced by Moscato [127] to introduce an extension to GAs where individuals in a population can be improved within a generation. Later, Moscato and Norman [128] presented an approach that applied a local search within a GA implementation.

Burke et al. [45] performed a hill climbing local search after each mutation instead of using a crossover operator. Light and heavy mutation operators were proposed to reassign single exams and sets of exams. However, none of the mutations were capable of improving the quality of the solutions when they were used on their own. In addition, a comparison with approaches that relied on multi-start random descent local search showed that this approach obtained better results for the Nottingham capacitated exam timetabling problem. Further tests on the Toronto benchmark showed that this approach was outperformed by their previous approach presented in Burke et al. [35].

Burke and Newall [44] presented a paper which focused on heuristic decomposition of the timetabling problem. A memetic algorithm was then used to solve sub-problems. The results showed that using heuristics and local

search approaches with GAs obtained better results than using a GA only. A detailed description on designing Memetic Algorithms for timetabling problems can be found in Burke and Landa Silva [26].

Ant Algorithms

Ant algorithms are another subset of population based approaches, introduced by Dorigo et al. [77]. Initially, ant algorithms were applied to the Traveling Salesman Problem in [76, 77]. Eventually, Costa and Hertz [61] developed the ANTCOL system which investigated the use of ant algorithms in graph colouring problems. Although the results produced from the system did not match the best reported in the literature, Costa and Hertz stated that the results were quite satisfactory. Socha et al. [151] applied ant algorithms to course timetabling. When they compared ant algorithms to other local search techniques, it was found that ant algorithms produced the best solutions.

The basic ANTCOL developed by Costa and Hertz [61] was extended by Dowsland and Thompson [80] and applied to the Toronto benchmark datasets with the aim of finding the minimum number of time slots required to produce a feasible timetable. Overall, the improved ANTCOL has produced competitive results compared to the results obtained using other approaches.

Eley [83] investigated the use of two modified ant algorithms based on the Max-Min Ant System for course timetabling [151], and the ANTCOL algorithm for graph colouring problems [61]. Both algorithms were hybridised with a hill climber and a comparison was made. It was observed that the ANTCOL outperformed the Max-Min ant system and the results were

comparable with the best results in the literature.

Hybrid Approaches

More recently, the timetabling research turned into a new direction of hybridised methods which combine two or more of the techniques mentioned earlier. Casey and Thompson [55] implemented a GRASP algorithm in exam timetabling. They observed that better solutions could be obtained when a limited form of Simulated Annealing with a high starting temperature and fast cooling was used in the improvement phase. Kempe chain neighbourhoods [159] were employed to the exams that contributed to the cost. Furthermore, a memory function was used to avoid exams sharing the same time slot as in the previous iteration.

Azimi [11] developed a hybrid heuristic based on a combination of an Ant algorithm and Tabu Search. This hybridisation was based on an earlier analysis where Simulated Annealing, Genetic Algorithms, Tabu Search and Ant algorithms were compared in [10]. The analysis was carried out with randomly generated exam timetabling datasets. Azimi introduced three different variants of hybridisation on Tabu Search and the Ant Colony method, and the results showed that the hybrid approaches work better than the meta-heuristics applied individually.

The hybrid approach developed by Caramia et al. [52] has produced several best known results in the literature on several instances of the Toronto benchmark datasets. They assigned the exams into the least number of time slots using a greedy scheduler followed by a penalty decreaser to improve the timetable. A penalty trader was applied to increase the number of time slots when no improvement could be made.

Merlot et al. [126] introduced another approach which also produced some of the best results in the literature. Their three-stage hybrid approach starts by constructing a feasible solution using a constraint programming approach. The solution is improved using Simulated Annealing in stage two, and in the final stage the solution is further improved by implementing a hill climbing method.

A study that investigated the hybridisation of large neighbourhood search based on improvement graph construction [4] and Tabu search was presented in Abdullah et al. [1]. A tree-based neighbourhood was implemented to perform cyclic exchanges among all of the time slots instead of considering the traditional pair-wise exchange based operators. Experimental results showed that their solutions were competitive to the best reported in the literature at the time of publication.

Asmuni et al. [7] used fuzzy logic to combine two out of three graph colouring heuristics. The idea was to combine the two heuristics into a single value which calculates the difficulty of allocating an exam to a time slot. The exams are ordered using this value and are scheduled in order. Furthermore, the approach was extended to tune the fuzzy rules instead of keeping them fixed [8]. This was achieved through automating the combination of construction and improvement techniques. They observed that the use of meta-heuristics to enhance solutions is dependent on the quality of the solution obtained during the construction process. The approach focused on combining multiple criteria to decide how to construct a solution. Fuzzy methods were used to combine three single construction heuristics into three different pair wise combinations of heuristics to determine the order in which the exams are inserted into the timetable. They presented a comparison on the performance of the various heuristic approaches with

respect to a number of criteria (overall cost penalty, number of skipped exams, number of iterations of a rescheduling procedure required and computational time).

Abdullah et. al [3] presented a hybridisation of an electromagnetic-like mechanism (EM) and the Great Deluge algorithm. The technique estimates the quality of the solution and calculates a decay rate at every iteration during the search process. These values depend on a force value calculation using the EM approach. This approach produced the best quality solution for four of the instances.

Burke et al. [32] introduced an approach which combines a variable neighbourhood search (VNS) with a GA which produced the best quality solution for one of the Toronto instances. Different initialisation methods were used to introduce variants of the technique including a biased VNS and its hybridisation with a GA. They analysed a number of different neighbourhood structures and were able to demonstrate that the approach was highly effective although it requires a relatively large amount of computational time. In addition Burke et. al [28] proposed a method where a hill-climber compares the candidate solution with a solution produced a couple of iterations back instead of the current solution. This was called the "late acceptance criteria" and it produced the best quality solutions for two instances of the Toronto benchmark.

2.3 The Exam Timetabling Track of the Second International Timetabling Competition (ITC 2007)

2.3.1 Problem Description

The ITC2007 exam timetabling track could be considered as a complex and a more practical dataset in comparison to the Toronto benchmark. This is due to the larger number of constraints it contains. A full description of the problem and the evaluation function can be found in [124]. In addition, the characteristics which define the instances are summarised in table 2.2. The problem consists of the following:

- A set of time slots covering a specified length of time. The number of time slots and their durations are provided.
- A set of exams which should be allocated to the time slots.
- A list of the students enrolled in each exam.
- A set of rooms with different capacities.
- A set of additional hard constraints (e.g. exam X must be after exam Y or exam A must use Room R).
- A set of soft constraints and their associated penalties.

In comparison to the Toronto benchmark, the ITC2007 dataset has more than one hard constraint. The hard constraints are as follows:

- No student sits more than one exam at the same time.

- The capacity for each individual room should not be exceeded at a given period.
- Period lengths should not be violated.
- Additional hard constraints should be all satisfied.

The soft constraints violations are summarised as follows:

- **Two Exams in a Row** The number of occurrences where a student sits two exams in a row on the same day.
- **Two Exams in a Day** The number of occurrences where a student sits two exams on the same day. If the exams are back to back then this is considered as a Two Exams in a Row violation to avoid duplication.
- **Period Spread** The exams have to be spread a certain number of time slots apart.
- **Mixed Durations** The number of occurrences where exams of different durations are assigned to the same room.
- **Larger Exams Constraint** The number of occurrences where the largest exams are scheduled near the end of the exam session. The number of the largest exams and the distance from the end of the exam session are specified in the problem description.
- **Room Penalty** The number of times where certain rooms, which have an associated penalty, are used.
- **Period Penalty** The number of times where certain time slots, which have an associated penalty, are used.

Table 2.2: Characteristics of the ITC2007 dataset

Instance	Conflict Density	Exams	Students	Periods	Rooms	no. of Hard Constraints
exam 1	5.05	607	7891	54	7	12
exam 2	1.17	870	12743	40	49	14
exam 3	2.62	934	16439	36	48	185
exam 4	15.0	273	5045	21	1	40
exam 5	0.87	1018	9253	42	3	27
exam 6	6.16	242	7909	16	8	23
exam 7	1.93	1096	14676	80	15	28
exam 8	4.55	598	7718	80	8	21

2.3.2 Approaches for the ITC2007 dataset

A three phased approach was developed by Muller [129] to solve the problems in the ITC2007 exam timetabling track. The first phase consists of an Iterative Forward Search algorithm to find a feasible solution. Hill climbing is used to find the local optima in the second phase. Finally, a Great Deluge Algorithm is applied to further explore the search space.

Gogos et al. [99] proposed a method which used a GRASP. In the construction phase, five orderings of exams based on various criteria are generated. Tournament selection is used to select exams until they are all scheduled. A backtracking strategy using a tabu list is employed as required. In the improvement phase, Simulated Annealing is used. Finally, room allocations are arranged using integer programming in the third phase. The work was further improved in [100].

Atsuta et al. [9] used a constraint satisfaction solver incorporating tabu search and iterated local search. The solver differentiates between the constraints and their corresponding weights during computation to improve performance. De Smet [70] also incorporated local search techniques in a solver called Drools. Drools is an Open-Source Business Rule Management System (http://www.jboss.org/drools/).

Pillay [135] introduced a biological inspired approach which mimics cell behaviour. The exams are initially ordered using the saturation degree heuristic and scheduled sequentially in the available "cells" i.e. time slots. If more than one time slot is available, the slot which causes the least overall constraints violation is chosen. Rooms are chosen using the best fit heuristic. If a conflict occurs before all the exams are scheduled, the timetable is rearranged to lower the soft constraints violation. This is described as

cell division. If the overall soft constraint violation is not improved without breaking hard constraints, cell interaction occurs. The time slots are swapped in this process to remove hard constraint violations. The process continues until a feasible solution is achieved. Finally, the contents of cells having equal durations are swapped to improve the solution. This is called cell migration.

McCollum et al. [123] proposed a two phased approach where an adaptive heuristic is used to achieve feasibility during the first phase. The second phase improves the solution through the employment of a variant of the Great Deluge Algorithm.

2.4 Chapter Summary

This chapter has presented a detailed description of exam timetabling problems. As timetabling problems are tedious tasks to solve manually, a wide variety of approaches and algorithms have been applied to timetabling problems with the aim of developing computer-based automated timetabling systems. An overview of the two benchmarks used in this thesis was presented. Various approaches that have been implemented on both the benchmarks were also highlighted. In the next chapter, a background to the hyper-heuristic techniques utilised in the remainder of this thesis is presented, for the reader not familiar with hyper-heuristics.

Chapter 3

Hyper-Heuristics

3.1 Introduction

Hyper-heuristics can be defined as heuristics which choose heuristics, as opposed to searching for a solution directly [146]. The basic idea of hyper-heuristics has been around since the 1960s when Fisher and Thompson presented an algorithm which combines local job shop scheduling rules using a probabilistic learning technique [87, 88]. The learning technique was used to choose one of two heuristics which assign jobs to machines and therefore the approach can be classified as a hyper-heuristic.

For the past 10-15 years and until today, meta-heuristics have been used to solve many combinatorial optimisation problems in different fields. This allowed the in-depth scientific understanding of these problems. In addition, a range of new meta-heuristics was successfully developed to solve difficult problems and problem-specific systems were produced as a result. Designing and implementing such systems is expensive and very difficult for users to understand and maintain. Instead, some users decide to use

very simple heuristics, which lead to low quality solutions. This motivated research on the creation of systems, which would operate on a range of related problems with the "goal of raising the level of generality at which optimisation systems can operate" [51]. In practice, this motivates exploring systems which can operate over a range of different problem instances and even across problem domains, without expensive and time consuming parameter tuning, while still maintaining a certain level of solution quality.

The 'No Free Lunch' theorem [165, 166] states that all search algorithms would yield the same average performance for all the possible finite search problems on a given finite domain. This would suggest that it is not possible to develop a general search methodology for all optimisation problems. However, this does not state that it is not possible to develop a search methodology for a group or range of problems.

In educational timetabling, the algorithms are developed to target a specific set of problems and sometimes instances of the same problem. The algorithms would also need to be tailored to specific requirements according to the requirement of the stakeholders depending on their resources and constraints [6, 164].

The power of tailored algorithms falls in the use of domain knowledge which allows them to exploit the search space of the problem in hand. This knowledge can be used to intelligently guide and adapt the heuristic search. Therefore, humans develop heuristics which rely on the features of a problem domain, and this aids the heuristics to perform better on average than random search.

Hyper-heuristic research is concerned with building systems which can automatically guide and adapt a search according to the structure of the

problem in hand. This can be achieved through the exploitation of the structure of a problem, and the creation of new heuristics for that problem, or intelligently choosing from a set of pre-defined heuristics. In other words, hyper-heuristic research aims to automate the heuristic design process through automating the decision of which heuristics to employ to explore the solution space for a new problem. The aim here is to automate the heuristic design process and produce optimisation tools available to stakeholders who require quick and cheap solutions for their optimisation problems. Examples of such stakeholders could range from a school with a timetabling problem or a company with a space allocation problem, to larger scale problems such as a packing problem in a freight company or a job scheduling problem in a factory. It is often an expensive and time consuming process for them to consult a team of analysts to gather their requirement and employ a tailored heuristic, which would be very specific to their needs. A general system which automatically exploits the knowledge about a given problem and creates or chooses predefined heuristics would be applicable to a range of organisations, potentially decreasing the cost to all. However, there could be a trade-off between the generality of such a system, and the quality of the solutions produced. On the other hand, stakeholders looking for a cheap solution are not interested in the optimal as long as they obtain an accurate and quick solution for their problem. For example, consider a university that currently solves its timetabling problems by hand. This university may find that the cost of contracting a company to design a system to solve their problem, would be higher than the benefit they would get in terms of better timetables. However, the cost of purchasing a general system which can automatically find solutions without an expert being involved, may be lower than the resulting cost reduction to the university after using such a system. If the quality of the

solutions produced is satisfactory compared to the cost, then this would encourage more universities to take the advantage of using such systems which apply heuristic search methodologies. Hyper-heuristic research aims to automate the heuristic design process to produce "good-enough soon-enough cheap-enough" solutions for organisations with similar optimisation needs[51]. The broad aim is to design an algorithm for solving a range of problems or instances of a problem that is fast, reasonably comprehensible, trustable in terms of quality and reliable in terms of its performance across that range of problems [146].

There are two classes of hyper-heuristics, explained in sections 3.2 and 3.3. One class aims to generate heuristics from a set of components while the second class aims to intelligently choose heuristics from a set of predefined heuristics which have been previously developed. This can be seen as a framework to automate the process of predefined heuristic choice and hybridisation. It is this second class that is the focus of the work presented in this thesis.

3.2 Hyper-heuristics to Create Heuristics

The class of hyper-heuristics which aim to create a heuristic from a set of potential components is a fairly recent approach investigated in the literature. The heuristics created can be classified as 'disposable' heuristics which are created for a single and known problem instance, and cannot be applied to another unknown or hidden one. In contrast, the heuristic can be created in a manner which allows it to be re-used on different instances of the same problem class.

The true challenge here falls in the creation of a new heuristic or a variation of a previously created heuristic, which may obtain the best possible result for an instance. Human tailored heuristics are usually targeted to solve a class of instances, as it is inefficient to create a heuristic for every problem instance.

For example, the 'best-fit' heuristic is well-known to perform well on one dimensional packing problems [116] and is created to be generally used in solving any bin-packing instance. However, best-fit can be easily outperformed where piece sizes are defined over a certain distribution, and a heuristic is tailored to fit the specific requirement given that the piece sizes are known [37].

Therefore, if the heuristic design process is automated, a computer system could produce a good quality heuristic for an instance in a practical amount of time. This heuristic could even produce a solution that may be better than that which can be obtained by a human created heuristic. This is because it would have been created using specific knowledge of that instance. Automating the heuristic design process offers the chance to easily specify the range of instances that a heuristic will be applicable to, and then obtain that heuristic with minimal human intervention.

Poli, Woodward and Burke [141] employ genetic programming to evolve heuristics for bin packing. They observed that space was wasted while placing a piece in a bin. This left space which is a smaller than the smallest piece still to be packed. The structure within which their heuristics operate is based on matching the piece size histogram to the bin gap histogram.

Fukunaga [90] present an automated heuristic discovery system for the SAT problem. After analysing the problem and breaking down the structure into

component parts, an evolutionary algorithm to evolve human competitive heuristics was developed. It was also shown that specific human created heuristics from the literature, such as GWSAT and WalkSAT, can be represented from this component set. Fukanaga states that due to the large number of possibilities involved, the task of combining heuristics is a difficult one. However, using an automated system would make the task of creating new heuristics easier.

Bader-El-Din and Poli [12] observe that the approach of Fukunaga results in heuristics which are composites of those in early generations. Therefore, they present a different methodology to generate heuristics for SAT, which makes use of traditional crossover and mutation operators to produce heuristics which are faster to execute. Using four existing human created heuristics, a grammar which allows significant flexibility to create completely new heuristics is defined. Furthermore, the same team of researchers develop a hyper-heuristic system which evolves constructive heuristics for timetabling problems [13]. Since the heuristics generated are not re-usable, the system is considered as an online learning method which can obtain better results than other constructive algorithms.

Work presented by Keller and Poli in [113] presents a linear genetic programming hyper-heuristic for traveling salesman problems. The system evolves sequences of swap heuristics which swap two or three pairs of edges. The complexity of the evolved heuristics is then increased by adding loop and conditional components.

Ho and Tay employ genetic programming to evolve composite dispatching rules for the flexible job shop scheduling problem in [108, 155]. The dispatching rules are functions which assign a score to a job based on the state of the problem. When a machine becomes idle, each job in the machine's

queue receives a score. The job with the highest score in the queue is the next job to be assigned to the machine. Jakobovic et al. apply the same technique for the parallel machine scheduling problem [110].

Dimopoulos and Zalzala [75] evolve priority dispatching rules, based on the 'Montagne' rule, to minimise the total tardiness of jobs for the single machine scheduling problem. Although the function and terminal sets are relatively simple, the system evolves heuristics superior to the Montagne, ADD, and SPT heuristics. The function set consists of the four basic arithmetic operators and the terminal set contains five elements representing the global and local job information.

Recent work by Kumar et al. presents the first genetic programming system in which heuristics for a multi-objective problem have been automatically generated with a hyper-heuristic [120]. The system evolves heuristics for the bi-objective knapsack problem. The heuristic evolved iterates through the pieces still to be packed and evaluates them according to the profit and weight. When the evaluation returns a value which is greater than one, the iteration stops and that piece is packed. A similar approach is used for the bin packing problem in [36] as it uses a threshold to stop the iteration during evaluating pieces.

Oltean [130] presents a linear genetic programming hyper-heuristic, which generates evolutionary algorithms. A series of instructions that manipulate values in a memory array are represented in a standard individual. The approach was applied to function optimisation, the traveling salesman problem, and the quadratic assignment problem. The memory positions in the array are referred to as 'registers'. The evolutionary algorithm population is represented as the memory array and each member is stored in one register. The instructions which operate on the memory array are

represented as the genetic operators.

Tavares et al. [154] present another methodology for evolving an evolutionary algorithm for a function optimisation problem. They evolve the main components of a generic evolutionary algorithm, including initialisation, selection, and the use of genetic operators. The approach is demonstrated through an example of evolving an effective mapping function of genotype to phenotype.

Pappa and Freitas employ a grammar based genetic programming system to evolve rule induction algorithms for classification [133]. They state that the creation of algorithms may result in better algorithms than ones tailored by hand, which is one of the motivations behind hyper-heuristics. In addition, such algorithms reduce development costs and is another reason why hyper-heuristic research is potentially beneficial to smaller organisations.

Further work by Krasnogor and Gustafson has shown that there is a link between memetic algorithms and hyper-heuristics, where local searchers are evolved in an evolutionary algorithm by the self generation of memes [119]. They evolve a memetic algorithm with genetic programming for protein structure prediction [118, 119]. A grammar is defined to express groups of memes, each of which performs a local search at a given point in a genetic algorithm.

3.3 Hyper-heuristics to Choose Heuristics

In the majority of the work presented in the literature, the hyper-heuristic is provided with a set of predefined heuristics. These heuristics are usually known to have performed well in the problem domain in which they

are used. When using this type of hyper-heuristic approach, the hyper-heuristic is used to choose which heuristic, or sequence of heuristics, to apply, depending on the current problem state.

Furthermore, testing different heuristics on a given problem is very time consuming. It could take days or even months to test a certain approach on a single instance of a problem and not achieving the desired result. This proves that an exhaustive search is often computationally difficult to accept. Therefore, hyper-heuristics involve the design of intelligent systems that could decide the appropriate heuristics to be used to solve a certain problem. In addition, the strengths of the individual heuristics can potentially be automatically combined. However, for such an approach to be worthwhile, the combination should "outperform all the constituent heuristics" [148]. The hyper-heuristic explores the heuristic space instead of exploring the solution space. After choosing a certain heuristic to be used, this heuristic acts on the problem and the solution is evaluated using an objective function. Finally, the hyper-heuristic uses this evaluation to decide whether to accept the current heuristic or switch to a different one.

As described earlier, a meta-heuristic is usually applied directly to search the solution space. This means that the meta-heuristic is able to directly access the domain knowledge. A framework has been presented in [39, 63–66] where the hyper-heuristic operates at a higher level of abstraction without using any domain knowledge. The hyper-heuristic can only access the low-level heuristics but it has no knowledge about the problem being solved. This suggests that a designed hyper-heuristic could be re-used to be applied on different problem domains by replacing the set of low-level heuristics and the objective function. Figure 3.1 presents a diagram of a general Hyper-heuristic framework.

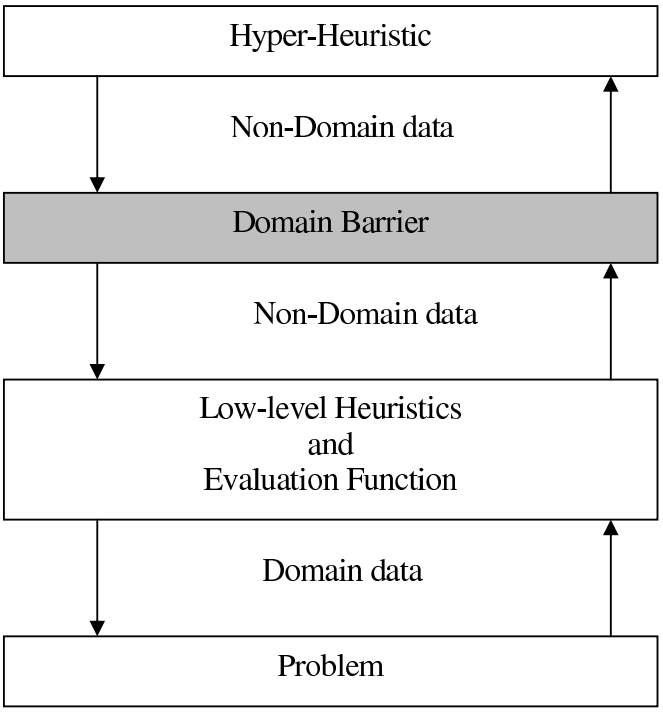

Figure 3.1: Hyper-heuristic Framework [39]

The hyper-heuristic can only access the non-domain data due to the presence of the domain barrier. Therefore the hyper-heuristic is not problem specific, as it has no knowledge of the problem being solved. It can only control the low-level heuristics that have access to the domain data and can directly act on the problem. The evaluation function then evaluates the solution and the results are passed to the hyper-heuristic which will take decisions accordingly.

As discussed in [39], there are two reasons for defining an interface between the hyper-heuristic and the low-level heuristics:

- Allowing the hyper-heuristic to communicate with the low-level heuristics using a standard interface instead of developing a separate interface for each low-level heuristic.
- Other domains could be easily implemented if the low-level heuristics

follow a standard interface. This would be done by supplying the hyper-heuristic with a set of low-level heuristics and the corresponding evaluation function.

The idea is demonstrated in a tabu search hyper-heuristic applied to nurse scheduling and university course timetabling in [25]. The same hyper-heuristic is used to act on two different sets of low-level heuristics for each of the two problems. The hyper-heuristic ranks the heuristics according to their performance and applies the heuristic with the highest ranking. If the heuristic does not obtain a better solution, then it is placed in a tabu list and subsequently not used for a number of iterations. The tabu search receives no information on the problem domain it is applied to, and therefore it could be used to solve a range of problems in different domains. A similar approach is applied to a real world timetabling problem from the MARA university of technology in [115]. An improved version of the approach was presented in [114].

Another example of a hyper-heuristic that maintains a domain barrier is the choice function hyper-heuristic [65]. At each decision point, the choice function evaluates the domain specific heuristics and chooses the best one. The choice function has three terms. The first is a measure of the recent effectiveness of the heuristic; the second is a measure of the recent effectiveness of consecutive pairs of heuristics; and the third measures the amount of time since the heuristic was last called. The first two factors make it more likely for good heuristics to be used more frequently. However, the third term adds the possibility of diversification. The approach obtained good results when applied to sales summit scheduling, presentation scheduling, and nurse scheduling. Applying the choice function on parallel hardware is investigated in [145].

Many existing meta-heuristics have been employed successfully as hyper-heuristics. A genetic algorithm and a learning classifier system have been applied to the one dimensional bin packing problem. The learning classifier system hyper-heuristic [149] selects between eight heuristics to pack each piece. The pieces remaining to be packed are assigned to four different ranges according to their size. The percentage of the pieces in each range in relation to the total number of pieces is calculated. The state of the problem is determined according to the result of the calculation and a rule is chosen to decide which heuristic to use to pack the next item. The system learns which heuristics should be used when different features are present. Subsequent work uses a genetic algorithm hyper-heuristic to evolve similar rule sets [148]. This work shows that good results can be obtained by assigning a reward to rule sets only when the outcome of the packing is known. Terashima-marin et al. apply a similar approach to the two dimensional stock cutting problem which involve a learning classifier system [156], and a genetic algorithm [157].

Work on a genetic algorithm hyper-heuristic for a trainer scheduling problem is presented in [67, 102–104]. The aim of the problem is to create a timetable of geographically-distributed courses over a period of several weeks using geographically distributed trainers [104]. The genetic algorithm chromosome represents a sequence of heuristics to apply to the problem rather than a timetable representing a solution. The work is extended in [104] to improve the genetic algorithm using an adaptive length chromosome, to encourage the evolution of good combinations of low-level heuristics without having to explicitly consider this optimal length. Then the results are further improved in [102] by guiding the genetic algorithm in adding and removing genes. A tabu method is added to the genetic algorithm in [103], to allow the genes which do not contribute to improving the

solution not to be used for a number of generations, instead of physically adding and removing genes.

Vazquez-Rodriguez et al. [160] employ another genetic algorithm on sequences of thirteen different dispatching rules to solve a multi-machine cardboard box shop scheduling problem. The hyper-heuristic was shown to be capable of learning effective hybridisations upon dispatching rules during scheduling, and thus was superior to employing single rules for the whole scheduling process.

Garrido and Riff [91, 92] also propose a genetic algorithm based hyper-heuristic for solving the 2-D strip packing problem. The hyper-heuristic is based on a set of four low-level heuristics. The hyper-heuristic uses a variable length representation which categorises the low-level heuristics according to their functionality: greedy, ordering and rotational. The number of objects to be positioned using each low-level heuristic is specified in the chromosome. Very good results are reported where the approach outperforms some of the specialised algorithms for the problem and benchmark instances presented.

An ant algorithm hyper-heuristic for the two dimensional bin packing problem is presented in [68]. The ant colony algorithm optimises a sequence of five heuristics, each with five parameters. An ant lays an amount of pheromone, proportional to the quality of the solution, on the path it takes to construct a full solution. Thus, the good sequences of heuristics become reinforced. Another ant algorithm hyper-heuristic is used in [38] for the project presentation scheduling problem. A standard ant algorithm is applied to a search space of heuristics by representing each heuristic as a node in a graph, where an edge between two nodes means one can be applied after the other. The ants traverse the graph, each producing a solution

using the heuristic associated with each node they travel through.

Simulated annealing is employed as a hyper-heuristic in [79] to solve the problem of selecting a set of shippers that minimises the total annual volume of space required to accommodate a given set of products with known annual shipment quantities. The simulated annealing algorithm is based on the tabu search hyper-heuristic presented in [25], where the heuristic is selected by the tabu search and is accepted according to the criteria defined in the simulated annealing algorithm. A simulated annealing hyper-heuristic is also employed in [16] for a shelf space allocation problem. The approach automates the design of planograms, which are used to show exactly where and how many facings of each item should physically be placed onto the store shelves. The performance of the simulated annealing hyper-heuristic was superior when compared to a greedy and choice function hyper-heuristic.

Bai et al. present further work inspired by a real world problem in [14], where a collaboration to consider a practical fresh produce inventory control and shelf space allocation model was tailored to suit a requirement from Tesco. Three hyper-heuristics are implemented for this problem, including the tabu search and simulated annealing hyper-heuristic previously presented in [79]. In addition, a comparison is made between meta-heuristic and heuristic approaches. A study on the effect of memory length on the performance of a simulated annealing hyper-heuristic is presented in [15].

Two genetic algorithm hyper-heuristics are investigated for the job shop scheduling problem in [78]. The first evolves the choice of a single priority rule from twelve rules to resolve a conflict at a given iteration. The other evolves a sequence of the sequence in which the machines are considered by a 'shifting bottleneck' heuristic. This is considered a hyper-heuristic

as the space of ordering heuristics is searched for one which minimises the makespan. Storer et al. present an approach for solving the job shop scheduling problem where they parameterise existing heuristics and search a space of parameters instead of solutions [152]. They state, "Search spaces can also be generated by defining a space of heuristics" [152], which is one of the main principles of hyper-heuristic research. Their subsequent research on how to successfully search such a space is presented in [153].

Cowling and Chakhlevitch [62] state that the performance of a hyper-heuristic relies very much on the choice of low-level heuristics. They address the question of how the set of low-level heuristics should be designed. Related to this is the problem of ensuring that the set of low-level heuristics is varied enough to ensure an efficient search, but not so large that it contains heuristics that are not necessary. Further work by the same authors addresses this problem, by introducing learning mechanisms into a hyper-heuristic to avoid using heuristics which do not make valuable contributions [57]. This reduces computational effort, because the under-performing heuristics do not then slow the search down.

A case based reasoning hyper-heuristic is presented in [41] for the course timetabling problem, and then again in [47], where it is shown that the hyper-heuristic can operate over both the exam timetabling and the course timetabling domains. This hyper-heuristic works by comparing the current problem to problems encountered before, and applying the same heuristics that have worked well in similar situations. Tabu search is employed in [24] to search for the best combination of two well known graph based heuristics for constructing exam timetabling solutions. The tabu search mechanism is based on that presented in [25]. The knowledge gained from this hyper-heuristic is then used to inform two hybrid graph based approaches to the

same problem, one which inserts a certain percentage of one heuristic into the heuristic list, and the other in which case based reasoning remembers heuristics that have been successfully used on similar partial solutions.

Pisinger and Ropke present a hyper-heuristic that can operate over five different variants of the vehicle routing problem [140]. The approach employs adaptive large neighbourhood search, a method which selects a heuristic to destroy part of the solution and a heuristic to rebuild it. Adaptive large neighbourhood search is a paradigm rather than a specific algorithm. It requires acceptance criteria to be specified as well as the heuristics which drastically modify the solution. The acceptance criteria used in this paper is simulated annealing, but any meta-heuristic could be used. The paradigm has similarities to variable neighbourhood search [105], which can also be considered a hyper-heuristic. This is because adaptively changing the neighbourhood of the search can be seen as intelligently selecting an appropriate heuristic.

Hyper-heuristics in Exam Timetabling

Ozcan et al. [132] state that one of the hyper-heuristic frameworks is based on a single point search in which heuristic selection and move acceptance are performed. They also highlight that most of the existing move acceptance methods compare a new solution to the current one to decide whether to accept it or not. They use a late acceptance strategy which compares between the new candidate solution and a previous solution that is generated a number of steps earlier. A set of hyper-heuristics utilising different heuristic selection methods combined with the late acceptance strategy were investigated on the Toronto benchmark.

Another approach was developed in [47] where a Case-Based Reasoning system was implemented. The system is used as a heuristic selector for solving exam timetabling problems. The different problem states and their corresponding constructive heuristics are stored in the system. The system then compares the problem to be solved with the cases that are already stored. The solutions are then constructed by repeatedly using the constructive heuristics suggested by the system.

Pillay and Banzhaf [138] present a genetic programming (GP) hyper-heuristic approach which acts over a search space of functions to assess the difficulty of allocating an exam during the timetable construction process. Each function is a heuristic combination of low-level construction heuristics combined by logical operators. The approach was applied to the Toronto benchmark on five instances of varying difficulty. The GP hyper-heuristic approach was found to generalise well over the five problems and performed comparably to other hyper-heuristic approaches combining low-level construction heuristics. In addition, a study to investigate the use of genetic algorithms (GAs) as a means of inducing solutions was conducted in [139]. They used a two-phased approach to the problem which focuses on constructing timetables during the first phase, while improvements are made to these timetables in the second phase to reduce the soft constraint violations. Domain specific knowledge in the form of heuristics was used to guide the evolutionary process.

The Graph based Hyper-heuristic (GHH)

In a graph based hyper-heuristic (GHH) for timetabling problems [142], a sequence of graph heuristics (see section 2.2.2) are used to order the events one by one to construct a full timetable. A number of high-level

heuristics are used to compare their performance to find the best heuristic sequences to solve exam and course timetabling problems. In each iteration, a heuristic sequence is chosen and used to construct a solution. After a sequence is applied and a solution is generated, this solution is evaluated using the objective function.

In each iteration of GHH, a timetable is constructed by adding the events one by one in the available time slots. According to the heuristic sequence being followed, the first event in the ordered list is scheduled to the first time slot that leads to the least cost. The cost in this situation is calculated by the soft constraint violations incurred. After an event is scheduled, the remaining events are re-ordered by using the next heuristic in the heuristic sequence. The process continues until all the events are scheduled and a full timetable is constructed.

An illustrative example of ordering five exams (e1-e5) using two graph heuristics is shown in figure 3.2. The example assumes that the exams are ordered according to their SD and LWD as shown. In the 1^{st} and 2^{nd} iterations, LWD is used (as shown in the heuristic sequence) and therefore e1 and e2 are scheduled. Assuming that the order of the remaining elements does not change after the SD list is updated in the 3^{rd} iteration, e5 is then scheduled. Finally, e4 followed by e3 are scheduled using the LWD ordering.

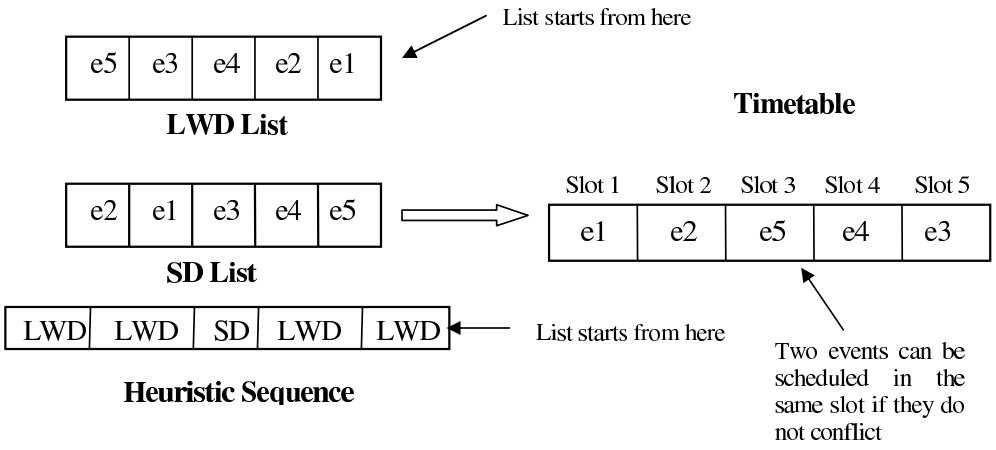

Figure 3.2: An illustrative example of using a heuristic sequence to construct a timetable

The quality of the timetable constructed is then evaluated and the cost is compared with any previous timetables produced using other sequences. Certain sequences during the construction of the timetable cannot schedule a certain event as it causes conflicts. Hence, the sequence is discarded and the high-level heuristic uses another sequence to construct another solution.

In this approach, the high-level heuristic acts on the heuristic sequences and does not act on the problem domain itself. The search is performed on the heuristic space and the high-level heuristic is controlled by the performance of running heuristic sequences to construct timetables. The GHH approach is used to construct a heuristic sequence which will then construct a solution. This technique applies heuristics adaptively to construct solutions and could therefore be generalised to be applied to a set of similar problems (i.e. timetabling problems in this case).

Qu et al. [143] presented a new direction in hyper-heuristic research where a random iterative graph based hyper-heuristic was used to generate a collection of heuristic sequences to construct solutions of different quality.

The sequences were a dynamic hybridisation of different graph colouring heuristics which was applied to the Toronto benchmark instances to construct solutions step by step. They observed that hybridisation of the LWD and SD heuristics tends to generate the best solutions. An iterative hybrid approach was developed based on these observations to adaptively hybridise LWD and SD at different stages of solution construction.

3.4 Chapter Summary

This chapter introduces hyper-heuristics which have been defined as heuristics which choose heuristics, as opposed to searching for a solution directly. Hyper-heuristics introduce the concept of automating the heuristic design process to provide more general and therefore cheaper systems. Two classes of hyper-heuristics are presented where one class aims to generate heuristics from a set of components while the second class aims to intelligently choose heuristics from a set of predefined heuristics which have been previously developed. In addition, previous work in the literature was discussed for each class. Furthermore, the hyper-heuristics approaches developed to solve exam timetabling which is the topic of this thesis was presented. Finally, the graph based hyper-heuristic which is used in chapters 4 and 5 is explained. In this thesis, we use different hyper-heuristics to either construct or improve solutions for a variety of exam timetabling problems. The next chapter presents a hyper-heuristic based on a GRASP to construct solutions for the Toronto benchmark.

Chapter 4

Adaptive Selection of Heuristics within a GRASP for Exam Timetabling Problems

4.1 Introduction

In the previous two chapters, we presented a review of the relevant literature regarding exam timetabling and hyper-heuristics. This chapter will present work on automating the selection of heuristics within a Greedy Randomised Adaptive Search Procedure (GRASP) to construct solutions for the Toronto benchmark. In the next chapter we will focus on tie breaking in heuristic orderings during the construction process.

For the past ten to fifteen years, meta-heuristics have strongly influenced the development of modern search technology at the inter-disciplinary interface of Artificial Intelligence and Operational Research [40]. Meta-heuristics can be applied in a wide variety of different areas such as schedul-

ing, data mining, cutting and packing, bio-informatics and many others. This facilitated the understanding of different problems in depth and the development of intelligent systems which would automatically solve large complex problems. Many of the meta-heuristic approaches which have been developed are particularly problem-specific. This is appropriate in certain circumstances but there are scenarios where we require a more generic methodology. This motivates the idea of designing automated systems that can operate on a range of related problems rather than on just one particular problem [39].

GRASP is a meta-heuristic which has attracted significant recent attention [86]. It is a two phase procedure which starts by adaptively creating a randomised greedy solution followed by an improvement phase. The aim of this work is to investigate if it is possible to use a GRASP to combine multiple heuristics simultaneously within a hyper-heuristic in order to provide a measure of the difficulty of placing each exam. If this is found to be possible then it would validate the hypothesis that a GRASP can be employed as a hyper-heuristic to combine heuristics. Experiments are performed to determine the effectiveness of the approach when applied to the Toronto benchmark.

In the following section we define some important terminology and present GRASP in detail. Section 4.2 discusses our intelligent adaptive hyper-heuristic using GRASP. The experimental settings and results are presented in section 4.3. Finally, a summary of the chapter is presented in section 4.4.

4.1.1 Greedy Random Adaptive Search Procedure (GRASP)

Randomly generated solutions are not expected to be of high quality. A greedy function, which does not contain ties or which contains deterministic rules to break ties, will always lead to the same solution. To overcome such issues a semi-greedy heuristic such as GRASP can be used [85, 106]. GRASP is a multi-start or iterative meta-heuristic which consists of two phases: construction and local search [85]. The construction phase builds a feasible solution which is improved in the local search phase. The process is performed for a certain number of iterations and the best overall solution is stored. At each iteration of the construction phase, a restricted candidate list (RCL) is created. The RCL is a set of all candidate elements that can be incorporated to the partial solution under construction without destroying feasibility. Two different methods were proposed by [85] and [106] to choose the elements to place in the restricted candidate list (RCL). The cardinality-based method chooses a number of elements from the top of the list. The size of the RCL is set through a parameter. The value-based method chooses the elements that fall within a certain threshold (0-100%) from the greedy value. More methods were also proposed in [106]. An element from the RCL is randomly selected and incorporated to the partial solution. The candidate list is updated and the next element for incorporation is determined by the evaluation of all candidate elements according to a greedy evaluation function.

GRASP can be easily integrated in a hybrid meta-heuristic, as any local search algorithm could be used in the improvement phase. Simulated Annealing and Tabu Search have been used in [60, 72, 121] as local search procedures.

In the next section, we present an adaptive approach where a graph based hyper-heuristic is used in the construction phase of a GRASP. The hyper-heuristic hybridises two low-level graph heuristics (SD and LWD) to obtain a heuristic sequence. A RCL is created and the heuristic sequence is used to fill the RCL. A heuristic is randomly selected from the RCL to order the unscheduled exams. Then, the exam on the top of the ordering is added to the timetable. After all the exams are scheduled, an improvement is made in the second phase of GRASP.

4.2 Methodology

4.2.1 The GRASP Construction Phase

Using a Graph based Hyper-heuristic to construct a Restricted Candidate List

In [142], a random ordering method was used in an iterative approach to generate heuristic sequences to be applied on exam and course timetabling problems. However, applying random heuristic sequences to a problem is usually a time consuming process. In this case, an approach that would always lead to a feasible solution would be preferred. This section explains how a GHH can be used to construct a RCL in a GRASP. The objective here is to create a RCL of heuristics instead of directly acting upon the problem to find a solution. Algorithm 1 presents the pseudo-code of the proposed technique to construct a RCL using a GHH in the construction phase of a GRASP.

Algorithm 1 The pseudo-code of the RCL construction in a GRASP using a GHH

u = number of unscheduled exams
create a heuristic sequence h_s of size u
$rclSize = 0.03 \times u$
create a restricted candidate list RCL of size $rclSize$
for $n = 0 \rightarrow rclSize$ **do**
  $RCL(n) = h_s(n)$
end for
Randomly choose a heuristic from the RCL
Apply the heuristic to order the unscheduled exams and insert the exam on top of the ordering in the timetable

It was observed in previous work [143] that heuristic sequences that combine SD with LWD perform the best for almost all the exam timetabling problem instances that were investigated. The motivation of combining LWD and SD comes from the fact that LWD identifies the difficulty of scheduling exams in relation to the rest and SD orders them according to the current state of the available time slots. Therefore, we developed a hyper-heuristic which generates a heuristic sequence using SD and LWD within the construction phase of a GRASP.

Creating a heuristic sequence to construct a RCL

In each iteration, the GHH starts by updating the SD and LWD orderings. The GHH then determines the size of the heuristic sequence required to construct the RCL. The size was initially set to 3% of the number of unscheduled exams. Different RCL sizes were tested and it was found that using a RCL which has a size from 1% to 3% of the unscheduled exams obtains the best solutions according to the size of the problem. Using a RCL of a large size increases the randomness in the solution construction and therefore the choice of the exam to be scheduled next becomes more

difficult. In contrast, using a very small RCL would limit the choice of exams and would not lead to a feasible solution. After determining the size of the RCL, the heuristics to be used are defined using the heuristic sequence. A heuristic is then randomly chosen from the RCL and the exams are ordered according to this heuristic. Finally, the exam on top of the ordering is scheduled.

The main role of the hyper-heuristic is to choose the heuristics required to construct the RCL. In previous work undertaken in [29, 42, 54], it was proved that using SD on its own outperforms other graph heuristics and results in a better outcome. This is due to the ability of SD to dynamically adapt itself to the current state of the solution construction as it orders events according to the available valid time slots. Therefore, we decided to let the GHH to construct a heuristic sequence which contains twice as many SD iterations than LWD. Hence, this technique gives the GRASP a higher chance to schedule the exams using SD. The size of the RCL will decrease adaptively as the exams are scheduled. In addition, the approach will automatically schedule the last 20 exams using SD only.

An illustrative example is given in figure 4.1 to use a GHH to construct a RCL. The GHH uses SD and LWD as low-level heuristics. The number of unscheduled exams are equal to 100. Assuming the parameter of the size of the RCL is set to 3%, a RCL of size 3 is created. The heuristics on top of the sequence are used to fill the RCL. A heuristic from the 3 is randomly chosen from the RCL. Assuming SD is chosen, the SD ordering is updated and e56 is scheduled. Since SD appears twice in the RCL, the chance of SD being chosen is twice as that of LWD.

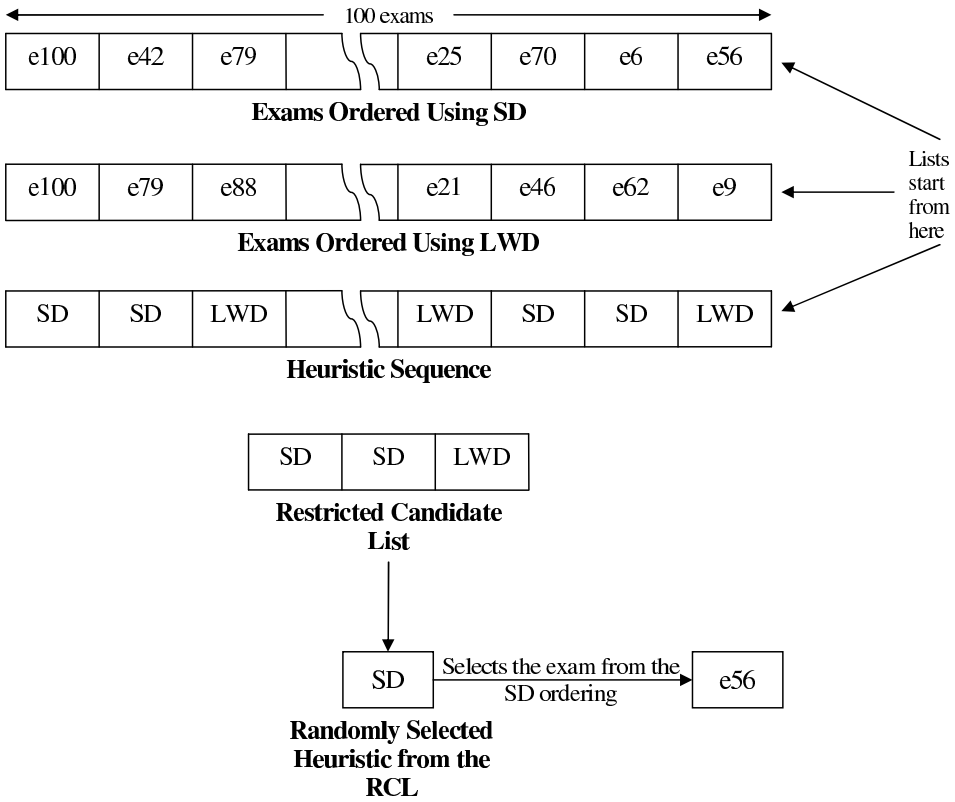

Figure 4.1: An illustrative example of building a RCL from SD and LWD

Adaptive selection of the Switching Point

Since SD orders exams according to the number of time slots available, starting to solve a problem using SD is inefficient as all the time slots would be available [29, 42, 54]. This would result in many ties at the beginning and different approaches could be used to overcome this difficulty. To avoid this problem, the GHH uses LWD only to construct the RCL and schedule the first exams in the first iterations. This process is done adaptively to find the switching point which leads to the best solution. The switching point is the point where SD is introduced into the heuristic sequence. The GHH switches to combine LWD and SD to build the GRASP RCL for the rest of the iterations. A random heuristic is then chosen from the RCL

and used to schedule an exam. This process will either schedule all the exams and construct a feasible solution or it will stop at a certain point where the exam cannot be scheduled in any time slot due to its conflict with those already scheduled. If a feasible solution is obtained, the solution is evaluated using the objective function. Otherwise, the process restarts if a feasible solution cannot be obtained.

In view of the fact that LWD orders the exams that have the largest number of students involved in clashes with other exams first, LWD gives priority to the exams that will be difficult to schedule later on in the process. Therefore, we used LWD to order exams in the first iterations before introducing SD in the heuristic sequence.

We applied the intelligent GHH to the Toronto benchmark exam timetabling problems [54] (see section 2.2) to adaptively select the best switching point from using LWD to combining both LWD & SD in a GRASP. The following formula was used to choose an initial switching point which is then adaptively tuned:

$$\text{Switching Point} = \frac{\text{Total Number of Exams}}{\text{Total Number of time slots}} \tag{4.2.1}$$

The starting switching point is the number of exams required so that the exams are spread evenly in the time slots available. The hyper-heuristic uses this as the initial switching point and applies the approach described in the previous section to obtain a solution. The switching point is then incremented and decremented by one and the new switching points are used to obtain another set of solutions. The process stops when a feasible solution is not obtained for two consecutive iterations.

Using this adaptive approach, the best switching point is automatically

found for different problems. We observed that using LWD on its own will not construct a feasible solution. Furthermore, SD has to be used early in the process to be able to identify exams that will be difficult to schedule later on. In addition, it is observed that using different switching points leads to different solutions. This is due to the fact that SD dynamically orders the exams and therefore the amount of LWD iterations employed at the beginning will affect the decisions made by SD.

4.2.2 The GRASP Improvement Phase

The GRASP improvement phase starts with a feasible solution from the construction phase and a steepest descent hill climbing improvement is employed. The hill climbing is run for $(e \times 10)$ iterations where e is the number of exams. In each iteration a random exam is chosen and the improvement of scheduling this exam in all the time slots of the timetable is observed. The best improvement is stored at the end of the process and the result is obtained. To summarise the whole approach, Algorithm 2 presents the construction and improvement phases of the GRASP.

Algorithm 2 The pseudo-code of the GRASP hyper-heuristic

```
for 1 → numberofiterations do
  while number of exams scheduled < total number of exams do
    Construct a RCL using a GHH (see Algorithm 1)
  end while
  if solution is feasible then
    Evaluate the solution and apply a steepest descent improvement
  else
    Discard the solution & continue without applying an improvement
  end if
  if current solution is better than the best solution then
    best solution = current solution
  end if
end for
```

4.3 Results

The adaptive GHH using GRASP was implemented and tested on a PC with an Intel Pentium 3.4 GHz processor, 1GB RAM and Windows XP. The program was coded using C++. The hyper-heuristic, the low-level graph heuristics and the GRASP are implemented as objects with a common interface to work together. The hyper-heuristic could be easily applied to different exam timetabling problems in the dataset. The objectives of the experiments are:

- To demonstrate the role of adding a GRASP to a GHH to improve the quality of the solutions obtained (shown in table 4.3).
- To raise the generality of exam timetabling problem solving approaches through producing a dynamic hyper-heuristic that would adapt to a given exam timetabling problem and generate results which are comparable to the results published in the literature (shown in table 4.3).
- To compare the quality of this approach with other known published methods (shown in table 4.3).

4.3.1 Analysis on the Intelligent Adaptive Hybridisation of the LWD and SD GHH to construct a RCL in a GRASP

In the first set of experiments, the GHH using GRASP is applied to version I and version II of the Toronto benchmark exam timetabling problems to see its ability to construct a feasible solution as well as to evaluate the quality of the solutions produced. The effect of intelligently hybridising different

low-level heuristics in a GRASP is investigated. It is observed that SD and LWD performed the best when hybridised together in the RCL rather than using them separately. The GHH chooses the exams which have a high priority in both the SD and LWD orderings to construct the RCL (i.e. the exams that have the largest number of students, involved in clashes with other exams and the exams that have the lowest number of slots available at that time to schedule in).

In the second set of experiments, using different percentages of SD and LWD in the RCL was investigated. It was observed that using the SD heuristic more than the LWD in the RCL produces better results. Therefore, we used a ratio of 2:1 SD to LWD heuristics to construct the RCL. This shows the importance of SD being used as it updates the ordering in each iteration and adapts to the current situation.

Five runs with 100 iterations each using distinct seeds are carried out for all the datasets except for (car92 I), (uta92 I), (uta92 II) and (car91 I) where only three runs were performed.The average costs of the solutions and the average percentage of feasible solutions constructed in the runs are presented in table 4.1. A comparison with using the same approach with SD and LWD separately is also shown.

Furthermore, another set of experiments is run to observe the effect of changing the size of the restricted candidate list used. Different RCL sizes were tested on problems with different size. The experiments showed that using a RCL with 3% the size of the unscheduled exams produces the best results for small problems (i.e problems with less than 200 exams). For medium sized problems, a RCL with 2% the size of unscheduled exams produced the best results (i.e problems with exams between 200 - 400). For large problems, a 1% RCL produced the best results (i.e. problems

Table 4.1: Average results, best results and percentage of feasible solutions produced from the GRASP construction phase using SD & LWD in comparison to using them separately. A (-) indicates that no feasible solution could be obtained

Problems	GRASP Construction using SD & LWD			GRASP Construction using only SD			GRASP Construction using only LWD		
	Average Cost	Best Cost	Percentage of feasible solutions	Average Cost	Best Cost	Percentage of feasible solutions	Average Cost	Best Cost	Percentage of feasible solutions
hec92 I	13.00	12.41	35%	13.82	13.02	21%	-	-	0%
hec92 II	13.72	12.37	20%	-	-	0%	-	-	0%
sta83 I	167.31	163.63	19%	174.61	169.10	15%	172.52	171.49	38%
sta83 IIc	40.42	36.82	69%	40.53	38.03	63%	36.03	34.79	69%
yor83 I	43.38	42.54	5%	45.62	44.21	4%	-	-	0%
yor83 IIc	52.57	52.24	3%	-	-	0%	-	-	0%
ute92 I	31.90	30.01	13%	32.78	30.74	10%	-	-	0%
ear83 I	39.06	37.71	32%	45.74	43.94	29%	-	-	0%
ear83 IIc	42.54	40.49	6%	48.27	48.21	3%	-	-	0%
tre92 I	9.38	9.00	39%	10.16	9.30	22%	-	-	0%
lse91 I	12.89	12.29	3%	13.48	12.51	2%	-	-	0%
kfu93 I	17.24	16.84	6%	18.28	18.24	2%	-	-	0%
car92 I	4.84	4.74	3%	5.07	4.85	0%	-	-	0%
uta92 I	3.65	3.62	100%	3.97	3.70	32%	-	-	0%
uta92 II	3.72	3.60	20%	-	-	0%	-	-	0%
car91 I	5.59	5.48	94%	5.78	5.49	57%	-	-	0%

with more than 400 exams). A large RCL introduces more randomness in large problems while a small RCL restricts the choice in small problems. In all the cases, the process starts with high randomness which adaptively decreases and becomes more deterministic as exams are scheduled.

Finally, another important factor for this method to construct a feasible solution is the switching point from using LWD to combining both SD and LWD in the GRASP. It is observed that for each problem, more than one switching point between the 1^{st} and the 25^{th} iteration led to a feasible solution. The choice of the switching point that would lead to the best solution is difficult especially when the method is applied to different problems. Using the formula and approach described in section 4.2.1, the switching point problem was adaptively solved. The average percentage of feasible solutions obtained in the runs for each problem is shown in table 4.1.

4.3.2 Analysis on GRASP improvement using steepest descent

In the second set of experiments, steepest descent is applied to the constructed solutions to perform a local search on the solution space. The improvement is applied to all the instances where a feasible solution is constructed. It is observed that the solutions with the least cost from the construction phase do not always lead to the best improvements. This is because the solutions constructed by different heuristic sequences in the heuristic space lead to different neighbourhoods in the solution space being searched to find the global optimum. We also investigated using tabu search and found that the tabu search improvement performs slightly better than steepest descent in some problems. However, the computational time for performing a tabu search improvement is much higher. Therefore, we used steepest descent as it produces competitive results in comparison with tabu search in a shorter time. Table 4.2 shows the average costs after the improvement and the total average time including construction and improvement.

Table 4.2: Average costs before and after improvement, best costs after improvement and total average time using Steepest Descent in GRASP

Problems	GRASP construction Average Cost	GRASP Improvement using Steepest Descent		Total Average Time (sec)
		Average Cost	Best Cost	
hec92 I	13.00	12.14	**11.78**	83.13
hec92 II	13.72	13.51	**12.04**	267.50
sta83 I	167.31	166.39	**158.94**	314.60
sta83 IIc	40.42	38.26	**35.19**	3376.87
yor83 I	43.38	42.85	**42.19**	484.58
yor83 IIc	52.24	51.53	**50.40**	410.77
ute92 I	31.90	31.14	**26.62**	402.50
ear83 I	39.06	38.71	**36.52**	5289.05
ear83 IIc	42.54	42.10	**39.93**	737.30
tre92 I	9.38	9.31	**8.99**	10817.01
lse91 I	12.89	12.72	**12.12**	768.33
kfu93 I	17.24	16.64	**15.45**	748.32
car92 I	4.84	4.67	**4.45**	8710.55
uta92 I	3.65	3.62	**3.50**	99053.51
uta92 II	3.72	3.69	**3.49**	95687.27
car91 I	5.59	5.48	**5.37**	99874.59

4.3.3 Comparison with state-of-the-art approaches

The best results obtained, the best reported in the literature using different constructive methods [144], the graph based hyper-heuristic [42], the tabu search hyper-heuristic [114] and other methods are presented in table 4.3. In addition, the standard deviation from the best reported results obtained is shown (σ in table 4.3).

The results obtained by this approach when compared to the state-of-the-art approaches, indicate the efficiency and generality of the intelligent adaptive hybridisation of LWD and SD GHH. In comparison with the pure GHH approach in [42], the GRASP-GHH performs better in 9 out of the 11 ver-

sion I cases. Furthermore, it performs better in 7 out of 8 cases reported in [114]. The results of (sta83 II), (yor83 II) and (ear83 II) were not compared to other results in the literature since the methods of modification to deal with the corruption (i.e. repetitions) in the datasets were not stated.

Although the best results reported in the literature were obtained by using different approaches that worked well only on specific instances, the results produced here using the adaptive approach are still in the range of the best-reported results as shown. In addition, the aim of this paper is to illustrate the adaptiveness and the effect of hybridising low-level heuristics in a hyper-heuristic using GRASP that could be applied to any timetabling problem with similar constraints. The objective here is not to outperform the other approaches that worked only on specific instances. However, we can demonstrate that our approach is competitive to the other approaches in the literature. It was not possible to compare the computational time as the time for most of the approaches was not reported in the literature.

Table 4.3: Best results obtained by the adaptive GHH using GRASP compared to the state-of-the-art approaches

Problems	Steepest Decent in GRASP		GHH Best [42]	Tabu search HH [114]	Constructive Best [144]	Best Reported [144]
	Best Cost	σ				
hec92 I	11.78	1.82	12.72	11.86	10.8	**9.2**
hec92 II	12.04	0.88	-	-	-	10.8
sta83 I	158.94	1.16	158.19	157.38	158.19	**157.3**
sta83 IIc	35.19	-	-	-	-	-
yor83 I	42.19	4.24	40.13	-	39.80	**36.20**
yor83 IIc	50.40	-	-	-	-	-
ute92 I	26.62	0.54	31.65	27.60	25.8	**24.4**
ear83 I	36.52	5.11	38.19	40.18	36.16	**29.3**
ear83 IIc	39.93	-	-	-	-	-
tre92 I	8.99	0.78	8.85	8.39	8.38	**7.9**
lse91 I	12.12	1.78	13.15	-	10.50	**9.6**
kfu93 I	15.45	1.73	15.76	15.84	14.00	**13.0**
car92 I	4.45	0.37	4.84	4.67	4.32	**3.93**
uta92 I	3.50	0.25	3.88	-	3.36	**3.14**
uta92 II	3.49	0.13	-	-	-	3.40
car91 I	5.37	0.62	5.41	5.37	4.97	**4.5**

4.4 Chapter Summary

In this chapter, we present an adaptive approach where two low-level graph heuristics (saturation degree and largest weighted degree) are dynamically hybridised in the construction phase of a greedy random adaptive search procedure (GRASP) for exam timetabling problems. The problem is initially solved using an adaptive LWD & SD graph hyper-heuristic which constructs the restricted candidate list in the construction phase of GRASP. It is observed that the SD heuristic is essential to construct a feasible solution. However, SD does not perform well at the early stages of the construction as the timetable is empty and it returns the same difficulty value for scheduling exams at the first few iterations. Therefore, LWD is used until a certain switching point. The switching point is adaptively tuned by the hyper-heuristic after evaluating the quality of the solutions generated.

An improvement is made to the constructed solutions and the best is chosen. Steepest descent is used for the improvement and it is shown to produce competitive results to a tabu search improvement in a shorter time. It is observed from experiments that the approach adapts to all the problems it is applied to and generates different quality solutions according to the switching point chosen. Therefore, we created an adaptive adjustment to the switching point according to the quality of the solutions produced.

The comparison of this approach with state-of-the-art approaches indicates that it is a simple yet efficient technique. It is also a more general technique than many in the literature. It can be adapted to construct good quality solutions for any exam timetabling problem with similar constraints.

Chapter 5

An Adaptive Tie Breaking and Hybridisation Hyper-Heuristic for Exam Timetabling Problems

5.1 Introduction

In previous work presented by Burke et al. [42], tabu search is used to search a set of heuristic sequences (rather than the solutions themselves) which consisted of the first five graph heuristics presented in section 2.2.2 and a random ordering strategy. In every step of tabu search, a sequence of these heuristics is used to construct a solution. Therefore, the search space of the tabu search consists of all the possible sequences of the low-level graph heuristics described in section 2.2.2. A tabu search move in this case creates a new sequence by changing two heuristics in the previous

heuristic list. A parameter is used to decide the number of exams scheduled in every iteration of the solution construction. The list is then applied to the problem in hand and the solution obtained is evaluated. If a better solution is obtained it is saved and the heuristic list is stored in the tabu list or otherwise it is discarded. The process continues for a number of predefined iterations depending on the problem size. The exams are scheduled to the first time slot that leads to the least cost. Algorithm 3 presents the pseudo-code of the approach. This approach and another approach developed in [143] can adaptively search and hybridise different low-level heuristics.

From these previously developed approaches, it was observed that Saturation Degree performs the best in most cases. The saturation degree of each unscheduled exam is calculated in each iteration and the exams are ordered according to the number of remaining time slots where an exam can be scheduled without violating any hard constraint. However, all the exams have the same saturation degree at the beginning of the solution construction as the timetable is empty. Ties could also occur anytime during the construction process when the same number of time slots is available for two exams. In this chapter, we analyse the effectiveness of applying different tie breakers to the Saturation Degree heuristic. In addition, we develop an intelligent adaptive approach which determines the heuristic to use in a hybridisation with Saturation Degree and the amount of hybridisation required to achieve the best solutions. The approach is tested on the Toronto benchmark as well as instances which are generated as part of this work.

The remainder of this chapter is structured as follows. Section 5.2 presents the instance generator we developed to test the approach. In section 5.3,

we present an investigation on the effectiveness of hybridisations and tie breaking in graph heuristics. We describe our approach and discuss our results in section 5.4. Finally, a summary of this chapter is presented in section 5.5.

Algorithm 3 The pseudo-code of the Tabu Search graph based hyper-heuristic [42]

```
Create a heuristic sequence h_s = {h_1, h_2, h_3 .... h_k} //k: number of exams/exams scheduled by each heuristic
for 1 → number of iterations do
  h = change two heuristics from the sequence h_s
  if h is not in the tabu list then
    Construct a solution using h
    if a feasible solution (s_c) is obtained & s_c < s //s: best solution so far
     then
      save the best solution, s = s_c
      update the tabu list
      h_s = h
    end if
  end if
end for
Output the best solution s
```

5.2 The Exam Timetabling Instance Generator

Different exam timetabling benchmark problems were collected and widely studied over the years due to the high-level of research interest in this area [144]. The data was collected from real world problems present in several institutions over the world. The benchmarks provide a means of comparison and evaluation to the different approaches developed. In addition to the Toronto benchmark described in section 2.2, the International Timetabling Competition (ITC2007) exam timetabling track [124] was re-

cently introduced (see section 2.3). It is a dataset which is more complex as it contains a larger number of constraints. As the focus of this work is on tiebreaking and hybridisations, the approach developed in this chapter was not applied to the ITC2007 dataset which requires different heuristics to schedule exams in rooms following certain constraints.

Since the Toronto benchmark is collected from real-world problems present in several institutions, they do not provide a full coverage of all possible problem scenarios. For example, the benchmark dataset does not contain instances with conflict density in the range from 19%-27% and 29%-42% (see table 2.1).

In this section, we introduce an exam timetabling instance generator and a benchmark exam timetabling problem dataset that consists of 18 different problem instances. The 18 problems consist of 9 small and 9 large problems. To be more precise, a problem is considered small if it does not contain more than 100 exams while a large problem exceeds 500 exams. The problems are generated with conflict density values starting from 6% to 47% using 5% intervals. The number of students and their enrolments are variables according to the problem size and conflict density. The problems have similar constraints and use the same objective function as the Toronto benchmark dataset stated in the previous section.

Table 5.1 presents the characteristics of the 18 problem instances in the dataset. They vary in several characteristics, not only with respect to the number of exams (ranging from 80 - 567), the conflict density (ranging from 6%-47%), but also the number of enrolments (ranging from 194 - 60168) and the number of time slots to assign the exams (ranging from 15-70). There are also different numbers of students (ranging from 66 - 15326) across different instances.

The dataset is available at *http://www.asap.cs.nott.ac.uk/resources/data.shtml* where each problem consists of 3 text files. The file names describe the size (SP for small problem & LP for large problem) and conflict density of the problem instances. The representation is extendable to add new characteristics. At the same website, we also provide a solution evaluator to evaluate the quality of the timetables produced. In addition, the website presents the instance generator we developed to produce the dataset.

We investigate the approach developed in this chapter by applying it to the Toronto benchmark exam timetabling problems and the dataset we generated in table 5.1. Our approach is tested on the newly developed dataset to highlight its generality on problems with a full range of conflict density. There is a large set of problem instances in the existing literature. However, none concerns the generation of a full range of problems for testing approaches which are dependent on problem characteristics.

Table 5.1: Characteristics of the Benchmark dataset produced by our problem instance generator

Problem	Exams	Students	Enrolments	Density	Time Slots
SP5	80	66	194	0.07	15
SP10	100	100	359	0.11	15
SP15	80	81	314	0.17	15
SP20	80	83	344	0.19	15
SP25	80	119	503	0.26	15
SP30	80	126	577	0.32	15
SP35	100	145	811	0.36	19
SP40	81	168	798	0.42	19
SP45	80	180	901	0.47	19
LP5	526	1643	5840	0.06	20
LP10	511	2838	10659	0.12	20
LP15	508	3683	14026	0.16	24
LP20	533	5496	21148	0.21	30
LP25	542	7275	27948	0.26	35
LP30	550	8798	34502	0.31	35
LP35	524	9973	38839	0.37	50
LP40	513	10826	42201	0.41	60
LP45	567	15326	60168	0.46	70

5.3 Methodology

5.3.1 Hybridising and Tie Breaking Graph Heuristics using the Conflict Density of Exam Timetabling Problems

As previously stated, many ties can appear in a Saturation Degree ordering of exams during solution construction. In previous work [29, 143], an exam is randomly chosen in such situations. The random choice in the earlier stages of the search has a great effect on the quality of the solutions produced. Therefore, a means of breaking these ties is essential. We initially developed a method of breaking ties in Saturation Degree using other graph heuristics. The Saturation Degree after breaking the ties is then hybridised with another heuristic in a sequence where random heuristic sequences with different percentages of hybridisation are generated. Each heuristic in the sequence is used to schedule a single exam.

This technique was applied to four instances (hec92 I, sta83 I, yor83 I and car91 I) of the Toronto benchmark exam timetabling problems described in table 2.1 for off-line learning of the best tie breakers and heuristic hybridisations leading to the best solutions for problems with different conflict densities. These instances were chosen as they vary in size and cover a range of conflict densities. The effect of tie breaking and hybridising different low-level heuristics on the quality of the solutions produced is analysed.

Algorithm 4 presents the pseudo-code of the random graph heuristic sequence generator. The approach starts by constructing different heuristic sequences which consist of two graph heuristics (Saturation Degree and one of the other heuristics described in section 2.2.2 using different pre-defined

percentages of hybridisation). The sequences are then applied to the instances to construct a solution. For each percentage of hybridisation, 50 heuristic sequences are generated using different seeds to construct solutions. Only the sequences that produce feasible solutions are saved and the rest are discarded (see Algorithm 4). According to the quality of the solutions obtained for each instance, the best tie breaker and low-level heuristic for hybridisation are identified. Further analysis is performed to find the relation of conflict density with the low-level heuristics used and the amount of hybridisation required to guide the search to the best solutions.

Algorithm 4 The pseudo-code of the random graph heuristic sequence generator using a tie breaker for Saturation Degree (tb:tie-breaker)

```
Set h_1 as the first graph heuristic to be used (SD, SD tb LWD, SD tb
LUWD, SD tb LD)
Set h_2 as the second graph heuristic to hybridise with h_1 (LWD, LD,
LUWD, LUD, CD)
for i = 0 → i = 1 do
  for n = 1 → n = 50 do
    Initialise the heuristic sequence h = {h_1 h_1 .... h_1}
    h = randomly change (i × size of h) heuristics in h from h_1 to h_2
    Construct a solution s_c using h
    if s_c is feasible & s_c < s then
      save the best solution, s = s_c
      save h
    else
      Discard the sequence h
    end if
    n = n + 1
  end for
  i = i + 0.05
end for
```

In work presented in the previous chapter and work undertaken in [29, 42, 54], it was shown that using Saturation Degree on its own usually outperforms other graph heuristics and results in a better outcome. Therefore, we investigated applying Saturation Degree without breaking ties as the primary heuristic in a sequence generator in one set of experiments. In

addition, we applied Saturation Degree while breaking the ties using some other heuristic orderings such as LWD, LD and LE in another set of experiments. Finally, the effect of hybridising Saturation Degree with different percentages of another graph heuristic (i.e. LWD, LD, LUWD, LUD, CD or LE) in both experiments was analysed.

5.3.2 Analysis of Saturation Degree Tie Breakers

Several heuristic orderings can be used to break ties that occur in the Saturation Degree list during solution construction. An illustrative example of breaking ties in a Saturation Degree list is shown in Figure 5.1. We assume that the exams are ordered according to their LWD and Saturation Degree as shown. We also assume that the saturation degree is the same for the first 3 exams (i.e. e1, e2 and e3) in the list. In this case, LWD is used to break the ties. The tied exams are re-ordered according to their position in the LWD list. Therefore, e2 is scheduled first and removed from both lists. The Saturation Degree list is then re-ordered again in the next iteration of solution construction.

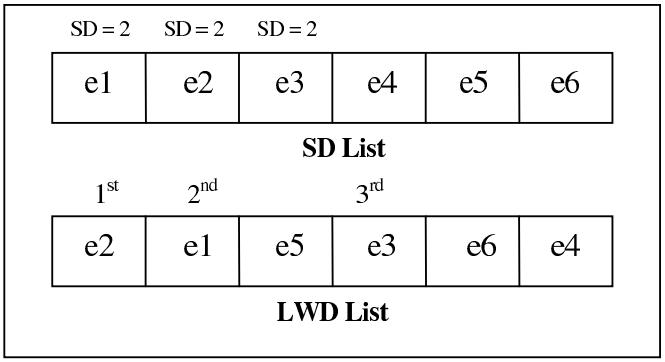

Figure 5.1: An illustrative example of breaking Saturation Degree ties

We applied several tie breakers to four instances of the Toronto benchmark exam timetabling problems [54]. This was run for 20 times on each instance

with different random seeds. Table 5.2 presents a comparison between the results obtained using Saturation Degree without breaking ties and Saturation Degree using different tie breakers.

It is observed that a Saturation Degree ordering without any tiebreakers does not produce a feasible solution when applied to HEC92 I and YOR83 I. After breaking the ties in Saturation Degree, a feasible solution could be obtained for all instances. The results also show that LWD is overall the best tie breaker for Saturation Degree as the exams are ordered according to their largest weighted degree when they have the same saturation degree.

A t-test is also carried out to give an indication if the results using the three different tie breakers are significantly different. Tables 5.3, 5.4 and 5.5 summarise the p-values of the t-tests carried out between the different tie breakers results. It can be seen that the results between the different tiebreakers are significantly different in all the cases except for hec92 I and sta83 I where LD and CD have no significant difference.

Table 5.2: Results using SD without tie breakers and with several different tie breakers. A (-) indicates that a feasible solution could not be obtained. The notation X tb Y denotes Y is used to break ties in X

	hec92 I	yor83 I	sta83 I	tre92
SD without breaking ties	-	-	178.24	9.68
SD tb LWD Best	**13.40**	**43.84**	**166.88**	**9.16**
SD tb LD Best	13.42	44.44	170.20	9.59
SD tb CD Best	13.67	46.78	168.21	9.19

Table 5.3: t-test on the results from breaking the ties using LWD and LD

	hec92 I	yor83 I	sta83 I	tre92
p-value	0.03	1.2E-04	0.01	6.71E-06
t Stat	2.14	6.25	2.59	7.05

Table 5.4: t-test on the results from breaking the ties using LWD and CD

	hec92 I	yor83 I	sta83 I	tre92
p-value	6.6E-03	6.42E-11	0.03	6.4E-06
t Stat	3.30	41.37	1.99	7.08

Table 5.5: t-test on the results from breaking the ties using LD and CD

	hec92 I	yor83 I	sta83 I	tre92
p-value	0.47	6.12E-08	0.27	0.02
t Stat	0.07	17.38	0.61	2.41

5.3.3 Hybridising Heuristic Sequences after Breaking the SD Ties

To analyse the effect of hybridisations, we decided to hybridise the SD ordering using a LWD tie breaker with different graph heuristics and to apply the sequences to the same four instances of the Toronto benchmark exam timetabling problems. Table 5.6 presents the results of applying these different hybridisations as well as a comparison against a hybridisation without using any tie breakers. As shown in table 5.6, hybridising the tie breaking SD ordering with other graph heuristics produced better results.

For problems where a feasible solution could not be obtained using SD without breaking ties (i.e. HEC92 I and YOR83 I), breaking the ties using LWD and a hybridisation using LD produced the best results. However, for problems where a feasible solution was obtained without handling ties in SD (i.e. STA83 I and TRE92), breaking the ties using LWD and a hybridisation using CD performed better. A possible reason could be the occurrence of many ties in the SD ordering which prevents it from producing a feasible

solution. Therefore, when tie breaking the SD ordering and hybridising it with LD, some exams will be ordered according to their number of conflicts with unscheduled exams. Generating a feasible solution using SD without handling ties is an indication that no ties or a very small number of ties occur. Therefore, applying a tie breaker and hybridising it with CD will order some examinations according to the conflicts with the exams already scheduled, producing better results than a LD hybridisation in this case.

A t-test is also carried out to give an indication if the results of using different hybridisations are significantly different. Tables 5.7, 5.8 and 5.9 summarise the p-values of the t-tests. It can be seen that the results between the different hybridisations are significantly different in all the cases except for hec92 I and tre92 when comparing hybridising SD tb LWD with LD and CD where no significant difference can be seen.

Table 5.6: Results of hybridising SD with other graph heuristics with and without breaking ties. The notation X tb Y denotes Y is used to break ties in X

	hec92 I	yor83 I	sta83 I	tre92 I
SD + LWD Average	13.02	44.59	170.08	9.25
SD + LWD Best	12.20	42.49	164.65	9.02
best % hybridisation	54	18	7	49
SD tb LWD + LD Average	12.60	43.03	168.21	9.03
SD tb LWD + LD Best	**11.89**	**42.16**	168.21	8.94
best % hybridisation	28	24	3	18
SD tb LWD + CD Average	12.69	43.34	163.32	9.00
SD tb LWD + CD Best	12.25	42.61	**159.50**	**8.83**
best % hybridisation	15	9	78	84

Table 5.7: t-test on the results from hybridising SD with LWD and SD tb LWD with LD

	hec92 I	yor83 I	sta83 I	tre92
p-value	2.19E-08	2.45E-08	4.83E-04	2.91E-10
t Stat	7.52	7.47	3.70	9.36

Table 5.8: t-test on the results from hybridising SD with LWD and SD tb LWD with CD

	hec92 I	yor83 I	sta83 I	tre92
p-value	3.54E-04	9.53E-07	1.55E-13	5.26E-10
t Stat	3.82	6.05	13.13	9.09

Table 5.9: t-test on the results from hybridising SD tb LWD with LD and SD tb LWD with CD

	hec92 I	yor83 I	sta83 I	tre92
p-value	0.14	0.003	1.03E-16	0.1
t Stat	1.11	2.98	17.75	1.33

5.3.4 Relating Conflict Density of Exam Timetabling Problems to SD Tie Breaking

By analysing the properties of the four problems and the results obtained using the different tie breakers and hybridisations, a relationship becomes apparent between the conflict density of exams in a problem and the percentage of hybridisations required (see Figure 5.2). As shown in table 5.6, the problems with conflict density of less than 25% (i.e. less than half of the exams are in conflict) obtained the best results when a hybridisation of less than 50% is used (i.e. more tie breaking SD is used than CD or LD). As the conflict density exceeds 25%, the percentage of the tie breaking SD used increases and the hybridised graph heuristic appears less in the sequences generating the best results. Having a higher conflict density means a higher probability of ties occurring in the SD ordering during the solution construction. Therefore, using SD and breaking the ties proves to be more effective. In contrast, instances with a lower conflict density will have a lower probability of ties occurring in the SD ordering during construction and using a lower percentage of the tie breaking SD is more

effective. The conclusions made here using four problems proved to be enough as they were verified later in section 5.4 after running the approach on the rest of the Toronto benchmark instances and the instances randomly generated.

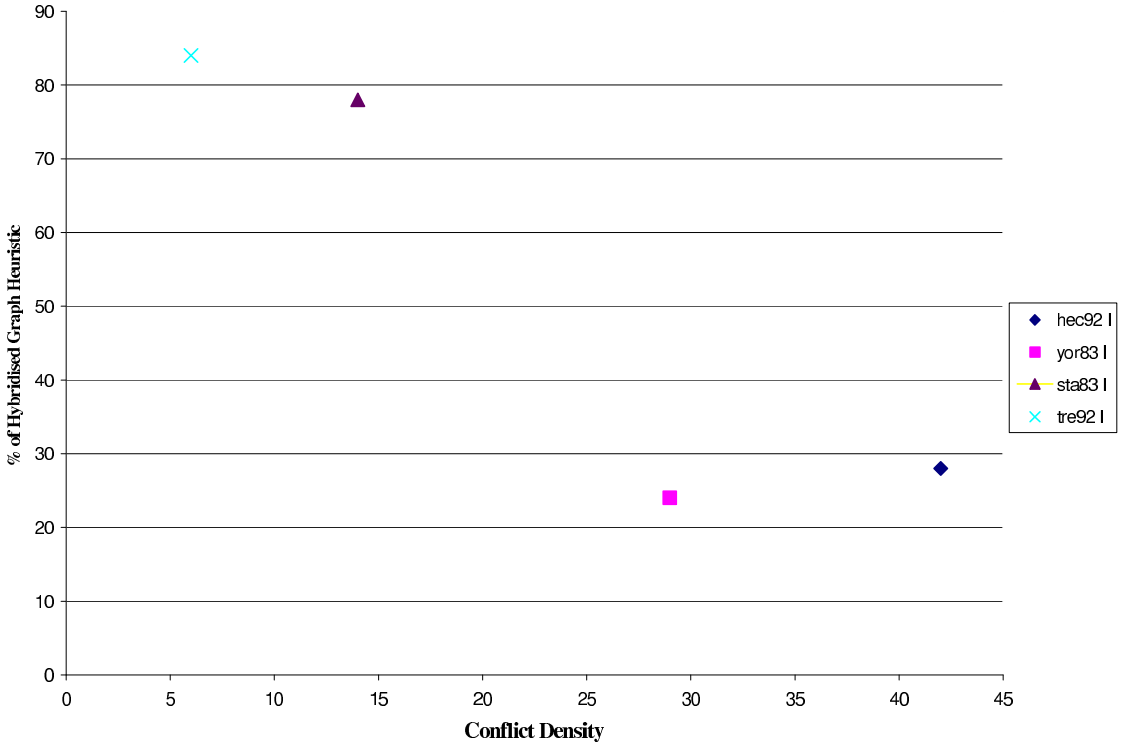

Figure 5.2: The relation between conflict density and the percentage of hybridisations obtaining the best solutions

5.4 Results

5.4.1 Adaptive Tie Breaking and Hybridisation for Benchmark Exam Timetabling Problems

The above observations indicate that there are two important factors to choose the most effective SD tie breaker and the percentage of hybridisation. The two factors are stated as follows:

- Whether a problem is solved using pure SD or not. If it is solved using pure SD then the tie breaking SD should be hybridised with CD. Otherwise it should be hybridised using LD.

- According to the conflict density of the problem, the percentage range of the tie breaking SD to be used could be defined.

Based on the above off-line learning we developed an approach which adapts to the problem in hand, choosing the most appropriate heuristic to be hybridised with the tie breaking SD and search for a solution within the percentage range of the tie breaking SD in a sequence. Figure 5 presents the pseudo-code of the Adaptive Tie Breaking (ATB) approach we developed.

Algorithm 5 The pseudo-code for the Adaptive Tie Breaking (ATB) approach

```
Schedule all exams using SD only
Set h1 to SD and set the tiebreaker as LWD //h1 : 1st heuristic in hybridisation
if a feasible solution could be obtained using SD only then
  Set h2 to CD //h2: 2nd heuristic in hybridisation
else
  Set h2 to LD
end if
if Conflict density > 25 % then
  i1 = 0.5
  i2 = 1
else
  i1 = 0
  i2 = 0.5
end if
for i1 → i2 do
  i = i1 // i: percentage of hybridisation
  for n = 1 → n = numberofexams × 10 do
    Initialise heuristic sequence h = {h1 h1 ... h1}
    h = randomly change (i × size of h) heuristics in h to h2
    Construct a solution sc using h
    if sc is feasible & sc < s then
      save the best solution, s = sc
      save h
    else
      Discard sequence h
    end if
  end for
  i = i + 0.05
  Save the smallest s and the corresponding h
end for
```

According to the conflict density of the problem, the range of percentages of hybridisations is defined. For problems with a conflict density greater than 25%, a percentage of 50% or greater of tie breaking SD is used. For problems with conflict density less than 25%, a percentage of 50% or less of tie breaking SD is used.

We tested this approach on the Toronto benchmark exam timetabling problems and present the results in table 5.10. Furthermore, we tested the approach on the dataset we developed to test the generality of the approach and present the results in table 5.11. The average computational time across the instances is also presented for 5 runs on a Pentium IV machine with a 1 GB memory. Note that this run time is acceptable in university timetabling problems because the timetables are usually produced months before the actual schedule is required [144].

The results obtained indicate the generality of our adaptive approach to all these exam timetabling instances regardless of the problem size. We compare our approach with other approaches where a random exam is chosen in the situation where a tie occurs. The results are presented in table 5.12. We also highlight the best results reported in the literature. In addition, the standard deviation from the best reported results obtained is shown (σ in table 5.12). Recall that the aim of this work is to illustrate the effect of breaking ties in heuristic orderings and automatically hybridising and adapting heuristics. We do not expect to outperform other heuristic and meta-heuristic approaches which are tailored specially for specific instances of this exam timetabling benchmark. However, we demonstrate that we can achieve results that are competitive with the best in the literature.

In comparison with the tabu search based hyper-heuristic in [114], our tie breaking approach performs better in 8 out of 11 cases reported. Furthermore, it outperforms in all the cases in comparison with the pure GHH investigated in [42]. Only the problems presented in table 5.12 were compared to other results since the others were not reported in the literature. Computational time was also not compared for the same reason.

Table 5.10: Results from the adaptive tie breaking (ATB) approach on the Toronto Benchmark dataset, Percentage of tie breaking SD (% of tb SD).Computational time is presented in seconds.

	ATB Average	% of tb SD Average	ATB best	% of tb SD best	Time (s)
hec92 I	12.61	75	11.89	85	274
yor83 I	43.03	71	42.16	81	1683
ear83 I	38.35	90	38.16	77	1475
sta83 I	163.33	42	159.50	56	357
car92	4.5	49	4.44	67	35812
car91	5.35	57	5.23	62	100304
uta92 I	3.64	52	3.50	53	54893
ute92	29.72	43	28.81	48	1469
lse91	11.89	47	11.73	39	2053
tre92	9.01	43	8.83	45	4293
kfu93	16.39	35	15.38	35	2697
hec92 II	12.52	69	11.84	73	346
yor83 II	51.24	91	50.40	84	1144
ear83 II	38.35	90	38.16	82	1457
sta83 II	36.10	47	35.00	42	1415
uta92 II	3.56	46	3.48	48	66339

Table 5.11: Results from the adaptive tie breaking (ATB) approach on the random dataset, Percentage of tie breaking SD (% of tb SD).Computational time is presented in seconds.

	ATB Average	% of tb SD Average	ATB best	% of tb SD best	Time (s)
SP5	3.71	34	3.55	34	10
SP10	11.91	48	10.54	45	30
SP15	17.15	37	15.56	35	29
SP20	20.33	44	18.69	41	33
SP25	25.18	54	23.22	52	43
SP30	33.89	52	31.56	60	78
SP35	47.42	64	45.19	74	141
SP40	28.58	53	27.28	61	109
SP45	32.76	78	31.08	81	100
LP5	9.27	42	9.12	39	4610
LP10	15.15	45	14.96	40	1803
LP15	13.31	29	13.05	24	2540
LP20	11.39	12	11.30	10	5708
LP25	9.93	89	9.90	86	16396
LP30	7.45	72	7.43	72	29367
LP35	6.81	90	6.76	91	40160
LP40	5.23	69	5.22	73	42306
LP45	4.85	72	4.77	81	59045

Table 5.12: Best results obtained by the adaptive tie breaking (ATB) approach compared to other Hyper-Heuristic approaches and the best reported in the literature

Problems	ATB Best	σ	GHH Best (Burke et al., 2007)	Tabu search HH (Kendall, 2005)	Best Reported (Qu et al., 2009a)
hec92 I	11.89	1.90	12.72	11.86	**9.2**
sta83 I	159.50	1.56	158.19	157.38	**157.3**
yor83 I	42.16	4.21	40.13	-	**36.20**
ute92 I	28.81	3.11	31.65	27.60	**24.4**
ear83 I	38.16	6.26	38.19	40.18	**29.3**
tre92	8.83	0.65	8.85	8.39	**7.9**
lse91	11.73	1.50	13.15	-	**9.6**
kfu93	15.38	1.68	15.76	15.84	**13.0**
car92 I	4.44	0.36	4.84	4.67	**3.93**
uta92 I	3.50	0.25	3.88	-	**3.14**
car91 I	5.23	0.52	5.41	5.37	**4.5**

To validate our results, we reversed the decisions taken by our approach in choosing a low-level heuristic and a percentage range of hybridisation, and applied the reversed approach to some of the problems with different characteristics and present a comparison in table 5.13. We present the average, best and standard deviation of the results obtained when using different combinations of the low-level heuristic and the percentage range used in the hybridisations. A paired t-test obtained a stat of 2.1, which is close to 2.11 ($p = 0.05$), demonstrating the adaptiveness of our approach during solution construction, indicating that hybridising the tie breaking SD with CD improves the quality of the solutions for problems solved using only SD. Otherwise a LD hybridisation is better. In addition, the comparison indicates that for problems with conflict density greater than 25% more tie breaking SD should be used than any low-level heuristic used in the hybridisation.

Table 5.13: A comparison of the results obtained by the adaptive tie breaking and the reverse of the approach

Problems	ATB Average	Reverse ATB Average	ATB Best	Reverse ATB Best	ATB σ	Reverse ATB σ
ute92 I	29.72	30.27	28.81	29.94	0.531	0.120
ear83 I	38.35	38.41	38.16	38.27	0.116	0.091
lse91	11.89	13.04	11.73	12.17	0.099	0.511
kfu93	16.39	17.13	15.38	16.90	0.589	0.142
car92 I	4.5	4.64	4.44	4.49	0.042	0.096
uta92 I	3.56	3.82	3.50	3.52	0.042	0.182
car91 I	5.35	5.51	5.23	5.24	0.076	0.165
SP5	3.71	4.24	3.55	3.82	0.099	0.252
SP10	11.91	11.98	10.54	10.78	0.797	0.702
SP15	17.15	17.71	15.56	15.98	0.924	1.008
SP20	20.33	20.52	18.69	18.89	0.953	0.950
SP25	25.18	25.21	24.17	25.18	0.589	0.030
SP30	33.89	34.22	31.56	32.02	1.351	1.279
SP35	47.42	47.85	45.19	46.00	1.293	1.077
SP40	28.58	29.57	27.28	27.61	0.756	1.140
SP45	32.76	32.86	31.08	31.18	0.976	0.979
LP5	9.27	9.35	9.12	9.29	0.093	0.046
LP25	9.93	9.99	9.90	9.96	0.025	0.03
LP45	4.85	5.03	4.77	4.78	0.053	0.154

5.5 Chapter Summary

This chapter presents an adaptive approach where a Saturation Degree heuristic, using Largest Weighted Degree to break ties, is dynamically hybridised with another low-level heuristic for exam timetabling problems. The hybridisation is performed according to the conflict density of the problem and the ability of the problem to be solved using a pure Saturation Degree (SD) heuristic. Largest Colour Degree First (CD) and Largest Degree First (LD) are used in the hybridisation process. CD is used for hybridisation if the problem is solved using a pure SD heuristic or LD is used otherwise. The amount of heuristic hybridisation is determined according to the conflict density of the problem. If the conflict density of a problem is greater than 25%, using more SD in a hybridisation with LD or CD and breaking the ties was more effective. On the other hand, for problems with conflict density less than 25% using more LD or CD than SD in a hybridisation was more effective. The approach is simple and performs the same regardless of the problem instance size although large instances take more time to solve. It performs better than a pure graph based hyper-heuristic and obtains reasonably good results when compared to a tabu search hyper-heuristic.

To test the generality of the approach and verify the results, an exam timetabling instance generator was developed and a set of benchmark exam timetabling problems was introduced and the first results on this data was reported. The generator takes the problem size and conflict density as inputs and generates a random problem with the required characteristics. To encourage scientific comparisons the generator and dataset is made available at *http://www.asap.cs.nott.ac.uk/resources/data.shtml*. The aim is to have a larger variety of exam timetabling problem instances with different

characteristics.

Chapter 6

A Hyper-heuristic using Improvement Low-level Heuristics for Exam Timetabling

6.1 Introduction

The previous two chapters presented hyper-heuristic approaches to construct solutions for exam timetabling problems. This chapter presents an approach to improve constructed solutions. In this case the low-level heuristics are used to perform certain moves to improve a constructed timetable. A random iterative hyper-heuristic approach which uses improvement low-level heuristics is presented. Several low-level heuristics can be used to improve a timetable with varying quality. The different low-level heuristics used could be considered as different methods for escaping from local

optima. However, the order in which exams are moved and the type of move performed play an important role in finding the best quality solution. An initial feasible solution is constructed using the Largest Degree heuristic where the exams in the ordering are assigned to the time slot causing the least penalty. In case there is more than one time slot with the lowest penalty, one of them is randomly chosen. Our objective is to analyse the performance of the different low-level heuristics used to minimise the penalty incurred from a constructed solution. In addition, we test the effect of using different orderings for the exams causing penalties in the solution. Finally we develop an adaptive approach which orders the exams causing violations and automatically selects the best heuristic to use for each exam to produce an improvement.

In the next section, the low-level heuristics used to improve solutions are presented, followed by a detailed explanation of the random iterative hyper-heuristic. Section 6.2.3 presents an analysis of hybridising improvement low-level heuristics, followed by an analysis on changing the ordering of exams causing a penalty in section 6.2.4. The proposed approach is described in section 6.2.5. Section 6.3 presents the results obtained when applying the approach to the Toronto benchmark and the ITC2007 dataset. Finally, a summary of this chapter is given in section 6.4.

6.2 Methodology

6.2.1 The low-level heuristics

In this chapter we investigate the effect of using different low-level heuristics or neighbourhood operators to improve timetables. A combination of two

improvement low-level heuristics is used in our approach. The following is a list of the heuristics investigated:

1. Move Exam (ME): This heuristic selects an exam and reassigns it to the time slot causing the least penalty.

2. Swap Exam (SE): This heuristic selects an exam and tries swapping it with a scheduled exam leading to the least penalty timetable.

3. Kempe Chain Move (KCM): This is similar to the SE heuristic but is more complex as it involves swapping a subset of conflicting examinations in two distinct time slots. This neighbourhood operator proved success in some previous research [32, 159].

4. Swap Time Slot (ST) : This heuristic selects an exam and swaps all the exams in the same time slot with another set of examinations in a different time slot. After testing all the time slots, the swap producing the least penalty timetable is applied.

6.2.2 The random iterative hyper-heuristic

The study presented in this chapter takes a similar approach to that presented in [143] where a random iterative hyper-heuristic was used to generate heuristic sequences of different quality to solve the Toronto benchmark. Instead of using the heuristic sequences to construct solutions, they are used here to improve constructed feasible solutions by rescheduling exams causing penalties. Algorithm 6 presents the pseudo-code of this random iterative hyper-heuristic. The process starts by constructing an initial feasible solution. Since the initial solution constructed affects the improvement

process, a random largest degree graph colouring heuristic which orders exams according to the number of conflicts each exam has with others is used [32]. This allows us to compare our approach to other approaches in the literature which use a similar method in construction. At every iteration, the exams causing violations in the constructed solution are identified and a random sequence of moves is generated. A move is the application of one of the low-level heuristics described in section 6.2.1. The sequence of moves is then applied to the sequence of exams as they are unscheduled one by one. Only moves that improve the current solution are accepted. If a move does not improve the solution, it is skipped and the exam stays in its current position. A sequence is discarded if an improvement is not obtained after the whole sequence is employed.

Algorithm 6 The pseudo-code of the random iterative hyper-heuristic with low-level improvement heuristics

```
order the exams using the LD heuristic
construct a feasible solution by assigning the exams to the time slot causing the least penalty
create a random ordered list of the exams contributing to the overall penalty incurred
for i = 0 → i = e × 50 //e: number of exams causing penalty do
  for n = 1 → n = e do
    initialise heuristic sequence h = [KCM KCM  KCM KCM] //h has size e
    h = randomly change n heuristics in h to ME, SE or ST
    construct a solution s_c using h
    if solution s_c is feasible then
      save h and the penalty of its corresponding solution s_c
    end if
  end for
end for
```

Figure 6.1 presents an illustrative example of the improvement process for a simple problem with six events (e1-e6) causing violations and ordered using SD. Assume that the sequence of six moves generated is "KCM ST KCM KCM KCM ST". An initial solution has been improved iteratively

by using the first two heuristics in the sequence. A swap time slot move is applied to exam e1 to swap all the exams in slot 1 with the exams in slot 2. In the next iteration, assuming that exam 5 is in conflict with exam 6, a Kempe chain move is applied to exam 5 where it is moved from slot 1 to slot 3 and exam 6 is moved to slot 1. The rest of moves are applied to the corresponding examinations in the sequence. In the case where an improvement is not achieved, the exam is kept in its current position. The sequences obtaining the best improvements are stored for the second stage where an adaptive approach is applied to generate more sequences which further improves the solution.

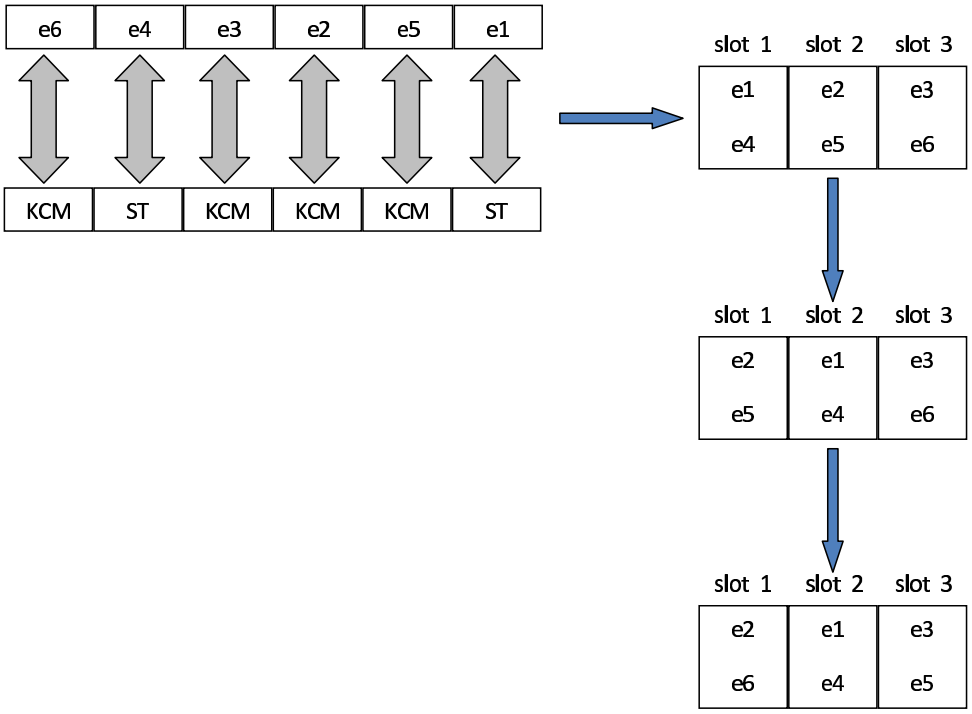

Figure 6.1: An illustrative example of solution improvement using a sequence of Neighbourhood Operators

This approach was applied to four instances (hec92 I, sta83 I,yor83 I and tre92) of the Toronto benchmark exam timetabling problems described in section 2.2 for off-line learning of the best heuristic hybridisations and the order of execution leading to the best improvement. These instances were

chosen as they vary in size and cover a range of conflict densities. After running this process for (ex50) times, where e is the number of exams causing soft constraint violations in the constructed solution, a set of sequences and the penalties of their corresponding solutions are obtained for further investigation on the effectiveness of the different heuristics used. Note that, only 50 samples were collected for each rate of hybridisation as we found that using more samples does not improve the quality of the final outcome at this point. Finally, an adaptive approach was developed and applied to the Toronto benchmark. Furthermore, to test the generality of the approach, it was applied to the ITC2007 exam timetabling track. The approach is presented in section 6.2.5.

6.2.3 Analysis of hybridising improvement low-level heuristics

In order to clearly observe the effect of the different low-level heuristics in improving solutions, the heuristic sequences generated consist of two heuristics. We use the Kempe chain move heuristic as the basic heuristic in the sequences as it has proved to be successful in previous work [32, 159]. The Kempe chain move involves swapping a subset of exams in two distinct time slots making sure that a hard constraint violation does not occur. The rest of the heuristics (ME, SE and ST) are randomly hybridised into the list of KCM.

The random sequences are generated with different percentages of hybridisation by inserting n ME, SE or ST, n = [1,..,e] in the sequences. For each hybridisation of KCM with either ME, SE or ST, 50 samples are obtained for each amount of hybridisation. Duplicate sequences are discarded and

another sequence is generated instead. The sequences are re-initialised in each iteration to avoid guiding the search and to explore a wider area of the search space at this stage.

We applied this approach to four instances of the Toronto benchmark exam timetabling problems [54]. Table 6.1 presents the results obtained using ME, SE and ST in a hybridisation with KCM as well as a comparison against using KCM only.

Table 6.1: Results using KCM without a hybridisation and with several different moves

	hec92 I	yor83 I	sta83 I	tre92
KCM without hybridisation Best	13.50	43.84	160.43	8.99
KCM with ME Best	12.03	43.84	157.48	8.91
KCM with SE Best	12.03	42.37	157.75	8.75
KCM with ST Best	**11.30**	**41.79**	**157.27**	**8.57**

It is observed that using a Kempe chain only produces the worst results. After introducing other heuristics in a hybridisation with the Kempe chain moves, better results were obtained. Another observation from table 6.1 is that swapping time slots and performing Kempe chain moves produces the best improvement for all the problems. One possible reason may be that swapping time slots allows the search to be more diverse and to sample different areas of the search space to find good solutions faster. In addition, no obvious trends could be obtained on the amount of ST hybridisation within the best heuristic sequences. However, it is observed in all the sequences leading to the best timetables that the ST heuristic is randomly distributed within the sequence and the percentage of hybridisation is less than 50%.

6.2.4 Variations of Orderings of the exams causing a penalty

To analyse the effect of ordering the unscheduled exams causing a soft constraint violation in a previous solution, we decided to test different orderings while using the Kempe Chain and swapping time slot hybridisation stated in the previous section. After the exams causing violations are identified, they are ordered first before being reassigned to a time slot. Several orderings can be used to help guide the search as follows:

- Largest Degree (LD) : The exams are ordered decreasingly according to the number of conflicts each exam has with others.
- Largest Weighted Degree (LWD) : The exams are ordered similarly to LD but weighted according to the number of students involved in the conflict.
- Saturation Degree (SD) : The exams are ordered increasingly according to the number of remaining time slots available to assign them without causing conflicts. In the case where ties occur, LWD is used as a tie breaker. From the work presented in chapter 5, it was shown that SD produces the best results when LWD is used to break ties in the ordering.
- Largest Penalty (LP) : The exams are ordered decreasingly according to the penalty they incur in the current solution.
- Random Ordering (RO) : The exams are ordered randomly.

Table 6.2 presents the results of applying different orderings to the unscheduled exams, then running a random heuristic sequence of KCM and

ST to assign them in better time slots.

Table 6.2: Results of hybridising KCM with ST using different orderings of the exams causing a soft constraint violation. The notation X tb Y means heuristic Y is used to break ties in heuristic X

	hec92 I	yor83 I	sta83 I	tre92
KCM with ST + RO Average	11.99	42.63	159.74	9.02
KCM with ST + RO Best	11.60	41.33	158.46	8.66
KCM with ST + LD Average	12.69	42.10	163.32	9.00
KCM with ST + LD Best	12.50	39.69	159.50	8.66
KCM with ST + LWD Average	12.06	42.08	159.74	8.91
KCM with ST + LWD Best	11.39	39.69	157.76	8.64
KCM with ST + LP Average	12.15	42.09	159.52	8.85
KCM with ST + LP Best	11.32	39.69	157.49	8.56
KCM with ST + SD tb LWD Average	11.45	41.96	159.39	8.74
KCM with ST + SD tb LWD Best	**11.25**	**39.56**	**157.37**	**8.54**

Table 6.3: t-test on the results from ordering exams causing violations using SD and LP

	hec92 I	yor83 I	sta83 I	tre92
p-value	1.3E-05	5.27E-21	1.9E-18	1.33E-04
t Stat	5.96	74.66	50.35	4.73

Table 6.4: t-test on the results from ordering exams causing violations using SD and LWD

	hec92 I	yor83 I	sta83 I	tre92
p-value	1.58E-07	5.3E-18	6.85E-16	5.56E-08
t Stat	8.67	47	33.89	9.41

Table 6.5: t-test on the results from ordering exams causing violations using LP and LWD

	hec92 I	yor83 I	sta83 I	tre92
p-value	0.18	7.51E-04	1.58E-11	3.36E-09
t Stat	0.93	3.87	17.04	11.62

As shown in table 6.2, we found that using SD and breaking any ties in the ordering using LWD produced the best results. This is because SD orders the unscheduled exams according to the number of time slots available to assign them without causing conflicts. Therefore, the chances of moving exams at the top of the SD list and finding better time slots for them become higher. Ordering the exams according to the penalty they incur proved to be the second best ordering followed by LWD. LD and RO performed randomly when applied.

A t-test is also carried out to give an indication if the results using SD, LP and LWD are significantly different. Tables 6.3, 6.4 and 6.5 summarise the p-values of the t-tests carried out between the results of different orderings, which are significantly different in all the cases.

6.2.5 Adaptive Selection of Low-level Heuristics for Improving Exam Timetables

Algorithm 7 presents the initialisation stage of the adaptive approach. The exams causing a penalty are first identified and are unscheduled. They are then put in a list and ordered using SD. Random heuristic sequences are generated using KCM and ST to reschedule the examinations. The sequences are then applied to the ordered exams and the corresponding solutions are saved. Note that, only 10 sequences are generated for each rate of hybridisation to be able to adhere with the time limitation and compare our results with the best in the literature for the ITC2007 dataset. Furthermore, limiting the number of sequences generated in this stage makes it easier to analyse and observe any trends in the sequences generating the best results.

Algorithm 7 The pseudo-code of the initialisation stage of the adaptive hyper-heuristic with low-level improvement heuristics

```
order the exams using the LD heuristic
construct a feasible solution by assigning the exams to the time slot causing the least penalty
create a list of the exams contributing in the overall penalty incurred ordered by SD
for i = 0 → i = e × 10 //e: number of exams causing penalty do
  for n = 1 → n = e do
    initialise heuristic sequence h = {KCM KCM ... KCM KCM}
    h = randomly change n heuristics in h to ST
    construct a solution using h
    if solution c is feasible then
      save h and the penalty of its corresponding solution c
    end if
  end for
end for
```

The above observations indicate that the best solutions were obtained when ordering the exams causing violations using SD, and rescheduling them using either a Kempe-chain move or swapping time slots. It was also observed that the heuristic sequences producing the top 5% results used the same move for the majority of the exams (i.e. the same heuristic appears in the same position in more than 75% of the sequences). Therefore, we developed an intelligent approach that performs an analysis to the best 5% of the sequences produced to generate a new set of sequences. The new set of sequences obtained better results for all the problem instances. The adaptive approach was tested and showed to be effective and comparable with the best approaches in the literature.

Algorithm 8 presents the pseudo-code of the approach which hybridises ST with KCM in two stages. The process is presented as follows:

1. In the first stage, the best 5% of heuristic sequences are collected and analysed. If the same heuristic is used in the same position for more

than 75% of the heuristic sequences, then it is stored. Otherwise the position is kept empty. Note that, we also tested collecting more than 5% of the best sequences to analyse them. However, the behaviour was random since no trends were seen. Therefore, to guarantee the effectiveness of the approach, the heuristic was chosen at a certain position if it appears in 75% of the best 5% heuristic sequences collected.

2. In the second stage, the empty positions are randomly assigned as KCM or ST. n x 5 sequences for the large problems (uta92 I, uta92 II, car91 and car92) and n x 10 sequences for the small problems are generated, respectively. The generated sequences are then applied to the problem.

Algorithm 8 Adaptive generation of heuristic sequences hybridising KCM and ST

```
construct initial heuristic sequences // see Algorithm 7
collect the best 5 % of the heuristic sequences
for i = 0 → i < number of exams causing penalty do
  count = 0
  for j = 1 → j < number of sequences do
    if heuristicSequence[i][j] = KCM then
      count ++
    end if
  end for
  if count > 0.75 × number of sequences then
    finalHeuristicSequence[i] = KCM
  else if count < 0.25 × number of sequences then
    finalHeuristicSequence[i] = ST
  else
    finalHeuristicSequence[i] = empty
  end if
end for
n = number of empty positions × 5 for large problems or number of
empty positions × 10 for small problems
for i = 0 → i < n do
  for j = 0 → j < number of sequences do
    if finalHeuristicSequence[j] = empty then
      finalHeuristicSequence[j] = KCM or ST
    end if
  end for
  construct a solution using finalHeuristicSequence[j]
  if a better solution is obtained then
    save finalHeuristicSequence[j] and the penalty of its corresponding
    solution
  end if
end for
```

6.3 Results

6.3.1 The Toronto Benchmark Results

The approach was tested on the Toronto benchmark exam timetabling problems and the results are presented in tables 6.6 and 6.7. The average

computational time for each stage across the instances is also presented for 30 runs on a Pentium IV machine with a 1 GB memory. In addition, the number of exams causing a penalty is presented in the tables.

The best results stated in the literature are presented in table 6.8. These include the hybridisation of an electromagnetic-like mechanism and the Great Deluge employed by Abdullah et al. [3], the hill-climbing with a late acceptance strategy implemented by Burke et al. [28], the variable neighbourhood search incorporating the use of genetic algorithms used by Burke et al. [32] and the sequential construction method developed by Caramia et al. [52]. These algorithms are described in section 2.2.2.

Table 6.6: Results from the adaptive improvement Hyper-heuristic (AIH) approach on the Toronto Benchmark dataset

	hec92 I	yor83 I	ear83 I	sta83 I	car92	car91	uta92 I	ute92	lse91	tre92	kfu93
AIH Average	12.69	41.74	38.98	159.21	4.49	5.39	3.56	27.97	11.45	8.90	15.54
AIH Best	11.19	39.47	35.79	157.18	4.31	5.19	3.44	26.70	10.92	8.49	14.51
Time spent in first stage(s)	254	1227	1139	576	10286	74973	35149	178	1189	2065	2153
Time spent in second stage(s)	143	456	553	183	41954	22988	26135	641	277	2228	592
Number of exams causing violations	18	22	26	22	42	49	40	27	39	29	34

Table 6.7: Contd. Results from the adaptive improvement Hyper-heuristic (AIH) approach on the Toronto Benchmark dataset

	hec92 II	yor83 II	ear83 II	sta83 II	uta92 II
AIH Average	12.43	50.49	41.98	35.00	3.54
AIH Best	11.35	49.72	39.60	32.57	3.45
Time taken in first stage (s)	352	861	1517	1204	53966
Time taken in second stage (s)	146	513	1275	684	27350
Number of exams causing violations	23	19	30	17	53

Table 6.8: Best results obtained by the Adaptive Improvement Hyper-heuristic (AIH) compared to the best approaches in the literature on the Toronto Benchmark

Problems	AIH Best	Abdullah(2009) Best [3]	Burke(2008) Best [28]	Burke(2010) Best [32]	Caramia(2008) Best [52]
hec92 I	11.19	9.73	10.06	10.00	**9.20**
sta83 I	157.18	156.94	157.03	**156.90**	158.20
yor83 I	39.47	34.95	**34.78**	34.90	36.20
ute92	26.70	24.90	24.79	24.80	**24.40**
ear83 I	35.79	36.00	32.65	32.80	**29.30**
tre92	8.49	8.5	**7.72**	7.90	9.40
lse91	10.92	10.03	9.86	10.00	**9.60**
kfu93	14.51	**12.62**	12.81	13.00	13.80
car92	4.31	**3.76**	3.81	3.90	6.00
uta92 I	3.44	**2.99**	3.16	3.20	3.50
car91	5.19	**4.42**	4.58	4.60	6.60

The results obtained indicate the generality of our approach to different constructed timetables regardless of the size. We also make a comparison with other hyper-heuristics which produced the best results in the literature in table 6.9. In comparison with the graph-based hyper-heuristic in [42], our approach performs better in all the cases reported. In addition, it performs better in 8 out of 11 cases in comparison with the hyper-heuristics investigated in [138] and [142]. Finally, it performs better in 10 out of 11 cases compared to the tabu search hyper-heuristic investigated in [114]. Only the problems presented in table 6.9 were compared to other results since the results for the other instances in table 2.1 were not reported in the literature.

Table 6.9: Best results obtained by the Adaptive Improvement Hyper-heuristic (AIH) compared to other hyper-heuristics approaches in the literature on the Toronto Benchmark

Problems	AIH Best	Kendall(2005) Best [114]	Burke(2007) Best [42]	Pillay(2009) Best [138]	Qu(2009) Best [142]
hec92 I	**11.19**	11.86	12.72	11.85	11.94
sta83 I	**157.18**	157.38	158.19	158.33	159.00
yor83 I	**39.47**	-	40.13	40.74	40.24
ute92	**26.70**	27.60	31.65	28.88	28.30
ear83 I	**35.79**	40.18	38.19	36.86	35.86
tre92	8.49	**8.39**	8.85	8.48	8.60
lse91	**10.92**	-	13.15	11.14	11.15
kfu93	**14.51**	15.84	15.76	14.62	14.79
car92	4.31	4.67	4.84	4.28	**4.16**
uta92 I	3.44	-	3.88	**3.40**	3.42
car91	5.19	5.37	5.41	**4.97**	5.16

6.3.2 The International Timetabling Competition (ITC2007) Results

To test the generality of our approach, we applied it to the ITC2007 exam timetabling dataset. The initial solution is constructed by ordering the exams according to their saturation degree. The exams are assigned a random time slot in the situation where more than one time slot is available. After a feasible solution is constructed the Adaptive Improvement Hyper-heuristic was applied to the constructed solution. To allow a fair comparison with the reported competition results, the approach was run for the same amount of time using 11 distinct seeds for each instance. Table 6.10 presents the results we obtained in comparison with the best in the literature. The description of the approaches used for comparison is presented in section 2.3.2. We do emphasise that the objective here is not to beat the best reported results but to demonstrate the generality of our approach to different problems with different constraints. A dash in the table means that no feasible solution was obtained.

The Extended Great Deluge in [123] obtained the best results for 5 out of the 8 instances. However, the approach was run for a longer time as it was developed after the competition. In the competition, the best results for all the 8 instances were reported in [129] using a three phased approach. The GRASP used in [99] produced the second best results.

In comparison with other hyper-heuristic techniques, our approach was able to produce better results in only one instance when compared to the evolutionary algorithm based hyper-heuristic presented in [137]. However, it was stated that they did not adhere to the time limitation imposed by the competition.

In comparison to the Constraint Based Solver developed in [9], our approach performed better in 3 out of the 8 instances. The approach using the Drools solver in [70] obtained feasibility for only 5 instances. Our approach outperformed it as we were able to gain feasibility for all the 8 instances. This demonstrates the generality of our approach to solving exam timetabling problems. Finally, our approach performed better on 6 of the 8 instances in comparison with the biologically inspired approach proposed in [135].

Table 6.10: Best results obtained by the Adaptive Improvement Hyper-heuristic (AIH) compared to the best approaches in the literature on the ITC2007 dataset

Instances	AIH Best	McCollum(2009) Best [123]	Muller(2008) Best [129]	Gogos(2008) Best [99]	Atsuta(2008) Best [9]	De Smet(2008) Best [70]	Pillay(2008) Best [135]	Pillay(2010) Best [137]
Exam 1	6235	4633	**4370**	5905	8006	6670	12035	8559
Exam 2	2974	405	**400**	1008	3470	623	3074	830
Exam 3	15832	**9064**	10049	13862	18622	-	15917	11576
Exam 4	35106	**15663**	18141	18674	22559	-	23582	21901
Exam 5	4873	3042	**2988**	4139	4714	3847	6860	3969
Exam 6	31756	**25880**	26950	27640	29155	27815	32250	28340
Exam 7	11562	**4037**	4213	6683	10473	5420	17666	8167
Exam 8	20994	**7461**	7861	10521	14317	-	16184	12658

6.4 Chapter Summary

The study presented in this chapter implements a hyper-heuristic approach which adaptively adjusts heuristic combinations to achieve the best improvement on constructed timetables. An investigation is made on the low-level heuristics used and the order in which exams causing soft constraint violations are rescheduled. The analysis is performed on a set of four benchmark instances of differing difficulty in an off-line learning process. It is shown that, of the heuristics tried, the best to combine with Kempe chains is the time slot swapping heuristic. In addition, better solutions are produced when ordering the exams causing a soft constraint violation using Saturation Degree and breaking any ties with Largest Weighted Degree. Based on the output of the learning process, an adaptive approach which analyses and adjusts some randomly generated sequences is implemented and applied to the rest of the instances. Furthermore, the hyper-heuristic approach is applied to a more constrained dataset, and showed to produce very competitive results compared to other approaches in the literature on both datasets.

The next chapter presents an adaptive approach which uses bin packing heuristics to assign exams to time slots and rooms. This is the first such system where bin packing heuristics are used within a hyper-heuristic to solve exam timetabling problems. This represents further work towards one of the objectives of hyper-heuristic research and, indeed, this thesis, to raise the level of generality at which optimisation systems can operate.

Chapter 7

Adaptive Selection of Heuristics for Assigning Time Slots and Rooms in Exam Timetables

7.1 Introduction

The one dimensional bin packing problem consists of a set of pieces, which must be packed into the least number of bins. Each piece j has a weight w_j , and each bin has a capacity c. The objective is to minimise the number of bins used, where each piece is assigned to one bin only, and the weight of the pieces in each bin does not exceed c. Bin packing heuristics on their own are simple techniques where items in the problem are packed using a specific strategy to construct solutions. For example, by using Best Fit (see table 7.1), a piece in a bin packing problem is placed into the bin which

has the least space remaining. Bin packing heuristics can be used in exam timetabling problems to assign rooms with different capacities to exams. Hence, using a Best Fit heuristic, the exam is placed in a room which has the least remaining capacity. The overall strategy is to make use of the same heuristics used in bin packing to assign resources in other problems such as exam timetabling.

In this chapter, we present an adaptive approach which hybridises heuristics used in bin packing to assign time slots and rooms to exams during the construction of a timetable. It is based upon the observations and statistical analysis over a large number of different heuristic sequences obtained by a random iterative hyper-heuristic generator. A similar process to the one described in section 6.2.2 will be used in the approach developed in this chapter. However, more than two heuristics are used in the hybridisation and a different method to adapt the heuristic sequences is presented.

Table 7.1: Packing strategies used to assign a time slot and room in timetabling

Packing heuristics	Packing strategies that allocate a time slot and room in the problem
Best Fit (BF)	Puts the exam in the feasible time slot and room which have the least remaining capacity
Almost Best Fit (ABF)	Puts the exam in the feasible time slot and room which have the second least remaining capacity
Worst Fit (WF)	Puts the exam in the feasible time slot and room which have the largest remaining capacity
Almost Worst Fit (AWF)	Puts the exam in the feasible time slot and room which have the second largest remaining capacity
First Fit (FF)	Puts the exam in the lowest indexed feasible time slot and room
Last Fit (LF)	Puts the exam in the highest indexed feasible time slot and room

7.2 Methodology

7.2.1 A Random Iterative Time Slot and Room Assignment Hyper-heuristic

The approach presented in this chapter focuses on the assignment of exams to time slots and rooms. It takes a similar approach to the one presented in the previous chapter where a random iterative hyper-heuristic generates heuristic sequences for the ITC2007 benchmark problem mentioned in section 2.3. Instead of using the heuristic sequences to order the exams to construct solutions or to improve constructed solutions, they are used here to assign exams to time slots and rooms.

Algorithm 9 presents the pseudo-code of this random iterative hyper-heuristic. The process starts by ordering exams using the LWD heuristic which orders the exams in a descending order according to the number of conflicts each exam has with others. The exams are weighted according to the number of students involved in the conflict (see section 2.2.2). Although it was proven in previous research [43, 54] that using SD performs the best in most cases due to its ability to dynamically order the events according to the number of remaining valid time slots, we have chosen to fix the order of the exams by using a static ordering heuristic to be able to focus on the effect of the heuristics assigning time slots and rooms.

At every iteration, the exam at the top of the list is chosen and the corresponding heuristic from the generated sequence is used to assign a time slot and room to the exam. The heuristics used for the assignment are those used in the one dimensional bin packing domain as described in table 7.1. In cases where ties appear when using BF, ABF, WF or ABF, the

time slot and room leading to the least penalty is chosen. If more than one time slot and room combination lead to the least penalty, one of these time slot and room combinations is randomly chosen. However, this does not apply when using FF or LF since these heuristics aim to assign exams in the first or last feasible time slot and room found. The process stops and the sequence is discarded if a feasible solution could not be generated. After a certain number of steps, a set of heuristics are collected for further analysis on the characteristics of good hybridisations of packing heuristics.

Algorithm 9 The pseudo-code of the random iterative bin packing based hyper-heuristic

```
Create an ordered list O of all the exams using LWD
for i = 0 → e x 50, e :the number of exams do
  for n = 1 → e do
    initialise heuristic sequence h_1 = {BF BF .... BF BF}
    initialise heuristic sequence h_2 = {ABF ABF .... ABF ABF}
    initialise heuristic sequence h_3 = {WF WF .... WF WF}
    initialise heuristic sequence h_4 = {AWF AWF .... AWF AWF}
    initialise heuristic sequence h_5 = {FF FF .... FF FF}
    initialise heuristic sequence h_6 = {LF LF .... LF LF}
    for s = 1 → 6 do
      h_s = randomly change n heuristics in h_s to BF, ABF, WF, AWF,
      FF or LF
      construct a solution c by applying the heuristic sequence h_s to the
      exams in list O (see Figure 6.1)
      if solution c is feasible then
        save h and the penalty of its corresponding solution c
      end if
    end for
  end for
end for
```

In this work, to investigate the effect of hybridising different heuristics to allocate time slots and rooms to exams, we generate a large number of heuristic sequences which consist of the 6 bin packing heuristics described in table 7.1. Since the allocation of examinations depends on the hard and soft constraints of the time slots and rooms in the problem, it is essential

that we examine a number of different strategies in assigning the exams. Since one of the objectives of hyper-heuristics is to avoid using any domain specific knowledge, we are only interested in the quality of the solutions obtained from the different allocation strategies to guide our search.

We start by applying the 6 initial heuristic sequences to the problem and analyse the effect of these sequences without performing a hybridisation in section 7.2.2. In section 7.2.3, we use the random iterative hyper-heuristic described above to generate a large number of hybridised heuristic sequences and analyse the results after applying the sequences to the problem instances. Finally, we develop an adaptive approach based on our observations and analysis in section 7.2.4.

7.2.2 Analysis of the Initial Heuristic Sequences

We start by applying the initial heuristic sequences, which contain only a single heuristic, to 8 instances of the International Timetabling Competition (ITC2007) exam timetabling dataset presented in section 2.3. The aim is to be able to compare the results before and after hybridising heuristics in the sequence. Since the timetable is empty, many ties appear at the beginning of process. A random choice is made from the set of feasible time slots and rooms available. Therefore, the initial sequences were run for 10 distinct seeds for each instance and the average and best results were collected. Table 7.2 presents the results obtained by running the 6 heuristic sequences which consist of a single heuristic.

It is clear, from table 7.2, that none of the heuristics used performed the best for all the exam timetabling instances. However, varying solution quality was obtained when applying the heuristics to the instances. This is

Table 7.2: Best and average results obtained by using a single heuristic. A (-) indicates that a feasible solution could not be obtained

Instances	**BF Best**	**BF Average**	**ABF Best**	**ABF Average**	**WF Best**	**WF Average**	**AWF Best**	**AWF Average**	**FF Best**	**LF Best**
Exam 1	12938	13690	13139	13712	-	-	-	-	-	**12577**
Exam 2	**3598**	4636	4403	5278	4043	5194	-	-	4158	3709
Exam 3	**17586**	19690	20143	21558	20261	21040	-	-	20083	20308
Exam 4	**25165**	28776	44358	44358	-	-	-	-	-	-
Exam 5	4909	5829	4826	5726	5553	5673	-	-	**4778**	4844
Exam 6	32175	32175	32575	32575	-	-	-	-	**31480**	38740
Exam 7	22388	23483	25446	26982	24203	24874	-	-	**21314**	23331
Exam 8	15972	16231	16046	16338	17766	17766	-	-	**15740**	18141

due to the fact that the instances have different structures and constraints. This indicates that different allocation strategies are required to solve the different instances. In addition, the results show that a feasible solution could not be obtained for any of the instances when using AWF.

Table 7.3 presents the capacities of the rooms in each instance ordered in the way they appear in the problem. For the first instance, feasible solutions were generated using BF, ABF and LF only. One possible reason may be that the first room in each time slot is the largest and using FF, WF or AWF would start by filling this room. This leaves exams with a large enrolment difficult to schedule later on in the process. The best solution was obtained using LF.

It can be seen that BF performed the best for the second and third instances. The second best solution was achieved using LF and FF for the second and third instances respectively. This may be due to the fact that the performance of the heuristics vary depending on the position of the room with the largest capacity in each instance. In addition, delaying the use of the largest room leads to better solutions. For example, in the second instance the first room in each time slot has the largest capacity. Therefore, LF performed better than FF for this instance. On the contrary, FF performed better than LF on the third instance since the rooms with the largest capacity are at the end of each time slot. The same observation also applies to the performance of FF and LF in the rest of the instances where a feasible solution was obtained using the two heuristics. Furthermore, since the distribution of room capacity is uniform in each time slot in the third instance, a small difference can be seen in the results obtained using FF and LF. Since the fourth instance contains only a single room in each time slot, only BF and ABF could find a feasible solution to this in-

stance. Note that although the LWD heuristic is used to order the exams, this does not mean that the exams with the largest number of students which require large rooms are scheduled first since the degree of conflicts with other exams is also considered.

Table 7.3: Room capacity ordered as they appear in each instance

Instance	Room Capacity ordered as they appear in the problem
Exam 1	260, 100, 129, 60, 77, 65, 111
Exam 2	424, 219, 120, 100, 40, 60, 60, 40, 36, 30, 30, 25, 72, 40, 35, 40, 38, 30, 60, 30, 85, 110, 100, 80, 70, 80, 40, 50, 92, 58, 195, 400, 90, 110, 264, 50, 19, 27, 60, 40, 51, 30, 39, 50, 50, 14, 127, 143, 23
Exam 3	127, 77, 41, 101, 93, 93, 76, 30, 70, 545, 275, 24, 171, 44, 70, 78, 59, 49, 27, 20, 39, 182, 32, 65, 58, 56, 61, 28, 26, 44, 78, 120, 500, 18, 30, 100, 11, 800, 16, 40, 16, 14, 116, 42, 51, 39, 500, 60
Exam 4	1200
Exam 5	896, 500, 999
Exam 6	240, 90, 210, 210, 110, 110, 80, 1000
Exam 7	240, 90, 210, 210, 110, 110, 70, 75, 70, 110, 110, 35, 45, 45, 1000
Exam 8	260, 100, 129, 60, 77, 65, 111, 120

7.2.3 Analysis of the Hybrid Heuristic Sequences

The random iterative time slot and room assignment hyper-heuristic generates a collection of heuristic sequences by hybridising different rates of BF, ABF, WF, AWF, FF or LF as described in Algorithm 9. The idea is to generate a variety of sequences which contain heuristics in different positions to use different strategies to schedule exams. Since the order of exams does not change during the solution construction, the effect of the heuristic used to schedule each exam can be clearly observed.

Due to the amount of heuristic sequences that could be generated from

using a large number of different heuristics, 5 heuristic sequences are initialised each using BF, ABF, WF, FF and LF. AWF was discarded from the hybridisation since it could not obtain any feasible result in table 7.2. The random iterative hyper-heuristic then systematically hybridises n BF, ABF, WF, FF or LF, $n = [1, ..., e]$, in each sequence. For each amount of hybridisation for each sequence, 50 samples are obtained. Although it is very difficult to cover the whole range of heuristic sequences, our focus is to explore the different areas of the search space by generating sequences which contain different hybridisations. Table 7.4 presents the penalties of the best and worst solutions obtained by these sequences. The corresponding amounts of BF, ABF, WF, FF and LF in the sequences are also presented.

It can be seen, from table 7.4, that the hybridisations leading to the best solutions mainly consist of the heuristic which produced the best solution using the initial sequences. Furthermore, the overall hybridisation of each heuristic in the best heuristic sequences for different instances depends on the performance of the heuristics before the hybridisation. For example for the first instance, from table 7.2, LF performed the best followed by BF then ABF. Therefore, the best heuristic sequence for the first instance employs more LF followed by BF then ABF.

Another observation from table 7.4 is that the worst solutions were obtained when hybridising heuristics which performed poorly when used on their own.

The initial heuristics help the hyper-heuristic to gather knowledge about the nature of the problem. Using this information, the search can be guided and the hybridisation process can be adapted and automated to improve the quality of the solutions.

Table 7.4: Penalties of the best and worst solutions from the heuristic sequences and the amount of hybridisation of BF, ABF, WF, FF and LF

Instances	**Best**	**% of hybridisation**					**Worst**	**% of hybridisation**				
		BF	**ABF**	**WF**	**FF**	**LF**		**BF**	**ABF**	**WF**	**FF**	**LF**
Exam 1	7821	26	17	0	0	57	14358	19	38	31	9	3
Exam 2	4295	34	6	18	16	26	6827	8	43	29	18	2
Exam 3	15386	32	19	9	28	12	22149	6	28	11	7	48
Exam 4	23713	63	30	0	0	0	50288	12	63	18	3	4
Exam 5	4288	14	20	6	34	26	6239	39	8	43	4	6
Exam 6	29349	28	17	0	42	13	40340	5	19	37	13	26
Exam 7	9752	25	8	13	35	19	27417	13	31	24	14	18
Exam 8	15675	27	21	11	39	2	18599	3	18	22	5	52

7.2.4 Adaptive Hybridisation of Bin Packing Heuristics

The above observations indicate that although the heuristics to be hybridised can be identified from running the initial sequences, the level of hybridisation of each heuristic and their appropriate ranges vary a lot for different instances. In addition, running the initial sequences gives an indication of the nature of the problem from the quality of the solution produced. Therefore, an intelligent approach needs to be developed to adaptively choose the heuristics to be used and the amount of each heuristic in the sequence. The adaptive approach is applied to all the instances and shown to be effective in comparison to the best approaches in the literature.

The adaptive approach is a three staged approach which identifies the heuristics to use in the hybridisation. A sequence is then initialised with a certain amount of each heuristic. Finally, an iterative adjustment is made to the amount of each heuristic in the sequence. The process is presented as follows:

1. In the first stage, the 5 initial heuristics containing only BF, ABF, WF, FF or LF are applied to the problem and the results are collected. The heuristics are ranked in a descending order according to the quality of the solutions produced. Heuristics which cannot produce a solution are discarded. A sequence is initialised using the heuristic producing the best result in this stage.

2. Based on the results obtained from the first stage, an adjustment is made to hybridise the initial sequence with the rest of the heuristics. The level of hybridisation of each heuristic is based on the number of heuristics producing a feasible solution and the ranking of heuristics

from stage 1. According to the ranking of heuristics the level of hybridisation of each heuristic is defined. The level of hybridisation is proportional to the ranking. For example, if 2 heuristics obtain a feasible solution for a problem which contains 100 exams, 67 of the exams will use the 1^{st} heuristic in the ranking while 33 will use the 2^{nd}. The following equation is used to determine the level of hybridisation of each heuristic:

$$\frac{\mathbf{e}}{\frac{\mathbf{f^2 + f}}{\mathbf{2}}} \times \mathbf{r} \tag{7.2.1}$$

e : the number of exams in the problem

f : the number of heuristics obtaining a feasible solution

The ranking **r** is calculated using the following equation:

$$\mathbf{r = f - i + 1} \tag{7.2.2}$$

f : the number of heuristics obtaining a feasible solution

i : the ranking of the heuristic according to the quality of the solution produced. The heuristic producing the best solution has the highest ranking.

Algorithm 10 presents the pseudo-code of the first two stages of the approach.

Algorithm 10 The pseudo-code of the initialisation stages of the adaptive bin packing based hyper-heuristic

```
create an ordered list O of all the exams using LWD
initialise heuristic sequence s_1 = {BF BF .... BF BF}
initialise heuristic sequence s_2 = {ABF ABF .... ABF ABF}
initialise heuristic sequence s_3 = {WF WF .... WF WF}
initialise heuristic sequence s_4 = {FF FF .... FF FF}
initialise heuristic sequence s_5 = {LF LF .... LF LF}
x = id of the heuristic sequence obtaining the best solution
f = number of sequences from s_1 to s_5 producing a feasible solution
s_h = s_x
for i = 1 → f do
    n = (e/((f^2 + f)/2)) * (f − i + 1) // equation (1)
    s_h = randomly change n heuristics in s_h to the heuristic used in sequence s_i
end for
construct a solution c by applying the heuristic sequence s_h to the exams in list O (see Figure 6.1)
if solution c is feasible then
    save s_h and the penalty of its corresponding solution c
end if
```

3. Based on the heuristic sequence obtained from the second stage, an iterative adjustment is made to tune the levels of hybridisations of each heuristic over the whole heuristic sequence. The aim in this stage is to increase the levels of hybridisation of the best performing heuristics.

   Algorithm 11 presents the pseudo-code of the approach used to tune the levels of hybridisation in the sequence obtained from stage 2. The heuristics in the sequence are tuned starting with the best heuristic in the ranking. The level of hybridisation in the sequence is increased by 1% by randomly changing other heuristics. Note that if the same heuristic is used in a randomly chosen position, another position is chosen to guarantee that the level of hybridisation is increased. Depending on the size of the problem, a number of sequences are generated and applied to the problem. The level of hybridisation is only

increased if a better solution is obtained. When an improvement is not obtained, the level of hybridisation of the next heuristic in the ranking is increased. The process stops when all the heuristics used in the sequence are increased and an improvement is not achieved.

Algorithm 11 The pseudo-code of the adaptive tuning of the levels of hybridisations in a heuristic sequence

```
create an ordered list O of all the exams using LWD
f = number of sequences producing a feasible solution
for i = 1 → f do
  h_i = the heuristic with rank i
  while an improvement is achieved do
    s_h = randomly change 1% of the heuristics in the sequence to heuristic h_i
    construct a solution c by applying the heuristic sequence s_h to the exams in list O (see Figure 6.1)
    if solution c is better than the previous solution then
      save s_h and the penalty of its corresponding solution c
    end if
  end while
end for
```

7.3 Results

We evaluate our adaptive approach by applying it to the ITC2007 instances described in table 2.2. The approach is run for the same amount of time specified in the timetabling competition, on a Pentium IV machine with a 1 GB memory, to allow a fair comparison with the reported results. The results are presented in table 7.5.

Table 7.5 shows that the results produced by the adaptive approach are better than the random approach presented earlier in table 7.4. In addition, table 7.5 presents the results we obtained in comparison with the best in the literature. The description of the approaches used for comparison is presented in section 2.3.2. We do emphasise that the objective here is not

to beat the best reported results but to demonstrate the generality of our approach to the different problem instances and the ability of a hyper-heuristic to adaptively hybridise heuristics used in bin-packing to assign events (exams in our case) to time slots and rooms. A dash in the table means that no feasible solution was obtained.

The Extended Great Deluge in [123] obtained the best results for 5 out of the 8 instances. However, the approach was run for a longer time as it was developed after the competition. In the competition, the best results for all the 8 instances were reported in [129] using a three phased approach. The GRASP used in [99] produced the second best results.

In comparison to the evolutionary algorithm based hyper-heuristic presented in [136], our approach was able to produce better results in only one instance. However, it was stated that they did not adhere to the time limitation imposed by the competition.

In comparison to the Constraint Based Solver developed in [9], our approach performed better on all 8 instances. The approach using the Drools solver in [70] obtained feasibility for only 5 instances. Our approach outperformed it, gaining feasibility for all 8 instances. This demonstrates the generality of our approach to solving exam timetabling problems. Finally, our approach performed better on all instances in comparison with the biologically inspired approach proposed in [135].

Table 7.5: Best results obtained by the Room and Time slot Assignment Hyper-heuristic (RTAH) compared to the best approaches in the literature on the ITC2007 dataset

Instances	RTAH Best	McCollum(2009) Best [123]	Muller('2008) Best [129]	Gogos(2008) Best [99]	Atsuta(2008) Best [9]	De Smet(2008) Best [70]	Pillay(2008) Best [135]	Pillay(2010) Best [136]
Exam 1	5752	4633	**4370**	5905	8006	6670	12035	8559
Exam 2	1693	405	**400**	1008	3470	623	3074	830
Exam 3	14586	**9064**	10049	13862	18622	-	15917	11576
Exam 4	21491	**15663**	18141	18674	22559	-	23582	21901
Exam 5	3844	3042	**2988**	4139	4714	3847	6860	3969
Exam 6	28480	**25880**	26950	27640	29155	27815	32250	28340
Exam 7	5182	**4037**	4213	6683	10473	5420	17666	8167
Exam 8	13711	**7461**	7861	10521	14317	-	16184	12658

7.4 Chapter Summary

The study presented in this chapter implements a hyper-heuristic approach which adaptively hybridises bin packing heuristics to assign exams to time slots and rooms. An investigation is made on the low-level heuristics used and their performance on different problem instances. A random iterative hyper-heuristic is developed to hybridise the heuristics and generate a large number of sequences of differing quality. It is shown that, the hybridisations leading to the best solutions mainly consist of the heuristic which produced the best solution when used on its own. Furthermore, the overall hybridisation of each heuristic in the best heuristic sequences for different instances depends on the performance of the heuristics before the hybridisation. Based on these observations, an adaptive approach which chooses the heuristics to use and tunes the level of hybridisation of each heuristic is implemented. Furthermore, the hyper-heuristic approach is applied to the International Timetabling Competition (ITC2007) dataset and showed to produce very competitive results compared to other approaches in the literature.

Chapter 8

Conclusion

In this thesis hyper-heuristic methodologies have been investigated in an attempt to construct and improve solutions to exam timetabling problems and to evaluate the generality of the approaches developed. A hyper-heuristic is a heuristic that searches a space of heuristics, rather than a solution space directly. The goal of such research is to develop systems which are more general than those currently available [39]. Designing and implementing problem-specific systems is expensive and very difficult for users to understand and maintain. Instead, some users decide to use very simple heuristics, which lead to low quality solutions. This motivated the creation of systems, which would operate on a range of related problems and yield good enough solutions. Eventually this would decrease the development costs and make it much easier for users to maintain such systems. In other words, raising the generality of the systems developed through the use of heuristics to choose the most appropriate heuristics to solve a problem in hand [25].

Furthermore, testing different algorithms on each problem is very time

consuming. It could take days or even months to test a certain technique on a single problem and not achieving the desired result. This proves that an exhaustive search is often computationally difficult to accept. Therefore, hyper-heuristics involves the design of intelligent systems that could decide the appropriate heuristics to be used to solve a certain problem. The hyper-heuristic explores the heuristic space instead of exploring the solution space. After choosing a certain heuristic to be used, this heuristic acts on the problem and the solution is evaluated using an objective function. Finally, the hyper-heuristic uses this evaluation to decide whether to accept the current technique or switch to a different one.

This thesis has presented and analysed different hyper-heuristic approaches which automate the construction and improvement of solutions for different exam timetabling problems. The heuristics chosen by these approaches produce results which are competitive to those obtained by tailored approaches. In this chapter, a list of contributions drawn from this research is provided, followed by a brief outline of some possibilities for future research.

8.1 Summary of Contributions

8.1.1 A Hyper-heuristic Using a GRASP to Construct Timetables

In chapter 4, a Greedy Random Adaptive Search Procedure (GRASP) is used to construct solutions for exam timetabling problems. Two low-level graph heuristics, Saturation Degree (SD) and Largest Weighted Degree (LWD) are dynamically hybridised in the construction phase of the GRASP.

The problem is initially solved using an intelligent adaptive LWD and SD graph hyper-heuristic which constructs the restricted candidate list (RCL) in the first phase of GRASP. It is observed that the size of the RCL used in each iteration affects the quality of the results obtained. In addition, the SD heuristic is essential to construct a RCL which leads to a feasible solution. However, SD does not perform well at the early stages of the construction. Therefore, LWD is used until a certain switching point is reached. The hyper-heuristic adaptively determines the size of the RCL in each iteration and the best switching point after evaluating the quality of the solutions produced. In the improvement phase of GRASP, it is observed that tabu search slightly improves the constructed solutions when compared to steepest descent but it takes a longer time. The approach adapts to all the instances of the Toronto benchmark. The comparison of this approach with state-of-the-art approaches indicates that it is a simple yet efficient technique. The results also indicate that the technique could adapt itself to construct good quality solutions for any timetabling problem with similar constraints.

8.1.2 Adaptive Tie Breaking and Hybridisation of Graph Colouring Heuristics

Graph colouring heuristics have long been applied successfully to the exam timetabling problem. Despite the success of a few heuristic ordering criteria developed in the literature, the approaches lack the ability to handle the situations where ties occur. In chapter 5, an investigation on the effectiveness of applying tie breakers to orderings used in graph colouring heuristics is presented. An approach to construct solutions for exam timetabling problems is proposed after defining which heuristics to combine and the amount

of each heuristic to be used in the orderings. Heuristic sequences are then adapted to help guide the search to find better quality solutions. The approach is tested on the Toronto benchmark problems and is able to obtain results which are within the range of the best reported in the literature.

In addition, to test the generality of the approach an exam timetabling instance generator and a new benchmark dataset which has a similar format to the Toronto benchmark are introduced. The instances generated vary in size and conflict density. The publication of this problem data to the research community is aimed to provide researchers with a dataset which covers a full range of conflict densities. Furthermore, it is possible using the instance generator to create random datasets with different characteristics to test the performance of approaches which rely on problem characteristics. The first results for the benchmark and the results obtained show that the approach is adaptive to all the problem instances that are addressed. We also encourage the use of the dataset and generator to produce tailored instances and to investigate various methods on them.

8.1.3 A Hyper-heuristic Using Low-level Heuristic Moves to Improve Timetables

Chapter 6 presents a hyper-heuristic approach which hybridises low-level heuristic moves to improve timetables. Exams which cause a soft-constraint violation in the timetable are ordered and rescheduled to produce a better timetable. It is observed that both the order in which examinations are rescheduled and the heuristic moves used to reschedule the exams and improve the timetable affect the quality of the solution produced. After testing different combinations in a hybrid hyper-heuristic approach, the

Kempe chain move heuristic and time-slot swapping heuristic proved to be the best heuristic moves to use in a hybridisation. Similarly, it was shown that ordering the exams using Saturation Degree and breaking any ties using Largest Weighted Degree produce the best results. Based on these observations, an iterative hybrid approach is developed to adaptively hybridise the Kempe chain move and time slot swapping heuristics in two stages. In the first stage, random heuristic sequences are generated and automatically analysed. The heuristics repeated in the best sequences are fixed while the rest are kept empty. In the second stage, sequences are generated by randomly assigning heuristics to the empty positions in an attempt to find the best heuristic sequence. Finally, the generated sequences are applied to the problem. The approach is tested on the Toronto benchmark and the exam timetabling track of the second International Timetabling Competition, to evaluate its generality. The hyper-heuristic with low-level improvement heuristics approach was found to generalise well over the two different datasets and performed comparably to the state of the art approaches.

8.1.4 A Hyper-heuristic Using Bin Packing Heuristics for Time Slot and Room Assignment

Chapter 7 presents an adaptive approach which hybridises bin packing heuristics to assign exams to time slots and rooms. The approach combines a graph-colouring heuristic which selects exams with bin-packing heuristics to automate the process of time slot and room allocation for exam timetabling problems. The process starts by analysing the quality of the solutions obtained by using one heuristic at a time. Depending on the individual performance of each heuristic, a random iterative hyper-heuristic

is used to randomly hybridise the heuristics and produce a collection of heuristic sequences to construct solutions with different quality. Based on these sequences, the way in which the bin packing heuristics are automatically hybridised is analysed. It is observed that the performance of the heuristics used varies depending on the problem. Based on these observations, an iterative hybrid approach is developed to adaptively choose and hybridise the heuristics during solution construction. The overall aim here is to automate the heuristic design process, which draws upon an emerging research theme which is concerned with developing methods to design and adapt heuristics automatically. The approach is tested on the exam timetabling track of the second International Timetabling Competition, to evaluate its ability to generalise on instances with different features. The hyper-heuristic with low-level graph-colouring and bin-packing heuristics approach was found to generalise well over all the problem instances and performed comparably to the state of the art approaches.

8.2 Extensions and Future Work

We suggest here some research directions in order to extend the work presented in this thesis.

8.2.1 Using More Than Two Heuristics in Hybridisations

Chapters 4 and 5 explore different hyper-heuristic approaches to hybridise two graph colouring heuristics to construct exam timetables. A similar approach could be used to intelligently hybridise more different low-level

graph colouring heuristics. In addition, some new adaptive heuristics similar to SD could be designed and used in the hybridisations.

8.2.2 Improving the Constructed Solutions Using Meta-heuristics

The solutions constructed in chapter 5 could be further improved by applying a meta-heuristic approach. Therefore, it would be interesting to study the effect of different improvement meta-heuristics on exam timetabling problems. The objective would be choosing the meta-heuristic that would produce the best improvement and automating this process.

8.2.3 Designing Hyper-heuristics to Choose Heuristics Depending on Problem Characteristics

In chapter 5, the approach developed used the conflict density of the problems to choose the heuristics to be used in a hybridisation and the amount of each heuristic in the heuristic sequences. Future research directions include observing the effect of more problem characteristics such as the number of students and enrolments on the performance of the approaches developed.

8.2.4 Applying Improvements to Partial Solutions

For the approaches investigated in chapters 6 and 7, future research directions include performing improvements during the timetable construction stage. This would allow changing the time slot or room assignment of a scheduled exam to make room for another exam.

8.2.5 Combining Heuristics to Construct and Improve Solutions in the Same Approach

In chapter 7, LWD was used to order the exams to be able to analyse the effect of using different bin packing heuristics used for time slot and room allocation. However, the effect of using a different heuristic or hybridising heuristics as in chapters 4 and 5 can be investigated. The objective is to combine heuristics used to order exams with heuristics used to assign time slots and rooms.

8.2.6 Applying the Approaches Developed to Other Problem Domains

Another future research direction is applying the techniques developed to random exam timetabling datasets with different constraints and evaluating their ability to adapt to the problems being solved. In addition, the techniques developed could be applied to graph colouring and course-timetabling problems using the same low-level heuristics. Furthermore, different low-level heuristics can be used in the approaches to solve problems from different domains.

References

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

[2] S. Abdullah and E.K. Burke. A multi-start large neighbourhood search approach with local search methods for examination timetabling. In *The International Conference on Automated Planning and Scheduling (ICAPS2006), Cumbria, UK*, pages 334–337, 2006.

[3] S. Abdullah, H. Turabieh, and B. McCollum. A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems. In *Proceedings of the 6th International Workshop on Hybrid Metaheuristics (HM2009)*, volume 5818 of *Lecture Notes in Computer Science*, pages 60–72. Springer, 2009.

[4] R.K. Ahuja, O. Ergun, J.B. Orlin, and A.O. Punnen. A survey of very large-scale neighbourhood search techniques. *Discrete Applied Mathematics*, 123:75102, 2002.

[5] J.S. Appleby, D.V. Blake, and E.A. Newman. Techniques for producing school timetables on a computer and their application to other scheduling problems. *The Computer Journal*, 3:237–245, 1961.

[6] H. Asmuni, E. K. Burke, J. M. Garibaldi, and B. McCollum. A novel fuzzy approach to evaluate the quality of examination timetabling. In *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT'06)*, Lecture Notes in Computer Science, pages 327–346. Springer, 2006.

[7] H. Asmuni, E.K. Burke, J. Garibaldi, and B. McCollum. Fuzzy multiple ordering criteria for examination timetabling. In E.K. Burke and M. Trick, editors, *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, volume 3616 of *Lecture Notes in Computer Science*, pages 334–353. Springer, 2004.

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

[9] M. Atsuta, K. Nonobe, and T. Ibaraki. Itc2007 track 1: An approach using general csp solver. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.

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

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

[12] M.B. Bader-El-Din and R. Poli. Generating sat local-search heuristics using a gp hyper-heuristic framework. In *Proceedings of the 8th In-*

ternational Conference on Artifcial Evolution, Lecture Notes in Computer Science, pages 37–49, 2007.

[13] M.B. Bader-El-Din and R. Poli. Grammar-based genetic programming for timetabling. In *Proceedings of the IEEE Congress on Evolutionary Computation(CEC 09)*, pages 2532–2539, 2009.

[14] R. Bai, E.K. Burke, and G. Kendall. Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *Journal of the Operational Research Society*, 59(10):1387–1397, 2008.

[15] R. Bai, E.K. Burke, G. Kendall, and B. McCollum. Memory length in hyperheuristics: An empirical study. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CISched 07)*, pages 173–178, 2007.

[16] R. Bai and G. Kendall. An investigation of automated planograms using a simulated annealing based hyper-heuristics. In *Proceedings of The 5th Metaheuristics International Conference (MIC 03)*, 2003.

[17] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35:268–308, 2003.

[18] P. Boizumault, Y. Delon, and L. Peridy. Constraint logic programming for examination timetablingboizumault1996. *Journal of Logic Programming*, 26(2):217–233, 1996.

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

[20] D. Brelaz. New methods to colour the vertices of a graph. *Communications of the ACM*, 22:251–256, 1979.

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

[22] B. Bullnheimer. An examination scheduling model to maximize students study time. In E. Burke and M. Carter, editors, *Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling*, pages 78–91, 1998.

[23] E. Burke and Y. Bykov. Solving exam timetabling problems with the flex-deluge algorithm. In *Proceedings of the Practice and Theory of Automated Timetabling Conference(PATAT2006)*, 2006.

[24] E. Burke, M. Dror, B. McCollum, S. Petrovic, and R. Qu. Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. *The Next Wave in Computing Optimization and Decision Technologies*, 29(1):79–91, 2005.

[25] E. Burke, G. Kendall, and E. Soubeiga. A tabu search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.

[26] E. Burke and J. Landa Silva. The design of memetic algorithms for scheduling and timetabling problems. In Krasnogor N., E. Hart, and J. Smith, editors, *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, volume 166, page 289312. Springer, 2004.

[27] E. K. Burke, Y. Bykov, J. P. Newall, and S. Petrovic. Time-predefined local search approach to exam timetabling problems. *IIE Transactions on Operations Engineering*, 36(6):509–528, 2004.

[28] E.K. Burke and Y. Bykov. 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.

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

[30] E.K. Burke, Y. Bykov, and S. Petrovic. A multicriteria approach to examination timetabling. In E.K. Burke and W. Erben, editors, *Selected papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT'00)*, volume 2079 of *Lecture Notes in Computer Science*, pages 118–131, 2001.

[31] E.K. Burke, D. de Werra, and J. Kingston. Applications to timetabling. In J. Gross and J. Yellen, editors, *Handbook of Graph Theory*, chapter 5.6, pages 445–474. Chapman Hall/CRC Press, 2004.

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

[33] E.K. Burke, D.G. Elliman, P.H. Ford, and R.F. Weare. Specialised recombinative operators for the timetabling problem. In *Proceedings of the AISB (Artificial Intelligence and Simulation of Behaviour) Workshop on Evolutionary Computing (University of Sheffield, UK, 3rd-7th April 1995)*, Lecture Notes in Computer Science, pages 75–85. Springer, 1995.

[34] E.K. Burke, D.G. Elliman, and R.F. Weare. A university timetabling

system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education*, 27:1–18, 1994.

[35] E.K. Burke, D.G. Elliman, and R.F. Weare. A hybrid genetic algorithm for highly constrained timetabling problems. In L. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA95, Pittsburgh, USA, 15th-19th July 1995)*, page 605610. Morgan Kaufmann, San Francisco, CA, USA, 1995.

[36] E.K. Burke, M.R. Hyde, and G. Kendall. Evolving bin packing heuristics with genetic programming. In T. Runarsson, H.G. Beyer, E.K. Burke, J.J. Merelo-Guervos, D. Whitley, and X. Yao, editors, *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN 06)*, volume 4193 of *Lecture Notes in Computer Science*, pages 860–869, 2006.

[37] E.K. Burke, M.R. Hyde, G. Kendall, and J. Woodward. Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one. In *Proceedings of the 9th ACM Genetic and Evolutionary Computation Conference (GECCO 07)*, pages 1559–1565, 2007.

[38] E.K. Burke, G. Kendall, J.D. Landa Silva, R.F.J. O'Brien, and E. Soubeiga. An ant algorithm hyperheuristic for the project presentation scheduling problem. In *Proceedings of the Congress on Evolutionary Computation 2005 (CEC 05)*, volume 3, pages 2263–2270, 2005.

[39] E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search

technology. In F. Glover and G. Kochenberger, editors, *Handbook of Meta-Heuristics*, pages 457–474. Kluwer, 2003.

[40] E.K. Burke, J.D. Landa-Silva, and E. Soubeiga. Multi-objective hyper-heuristic approaches for space allocation and timetabling. In T. Baraki, K. Nonobe, and M. Yagiura, editors, *Meta-heuristics: Progress as Real Problem Solvers*, pages 129–158. Springer, 2005.

[41] E.K. Burke, B.L. MacCarthy, S. Petrovic, and R. Qu. Knowledge discovery in hyper-heuristic using case-based reasoning on course timetabling. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002)*, volume 2740 of *Lecture Notes in Computer Science*, pages 276–286. Springer, 2002.

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

[43] E.K. Burke and J. Newall. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of operations Research*, 129(2):107–134, 2004.

[44] E.K. Burke and J.P. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

[45] E.K. Burke, J.P Newall, and R.F. Weare. A memetic algorithm for university exam timetabling. In *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling(PATAT1996)*, pages 496–503, 1996.

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

[47] E.K. Burke, S. Petrovic, and R. Qu. Case based heuristic selection for timetabling problems. *Journal of Scheduling*, 9(2):115–132, 2006.

[48] E.K. Burke, R. Qu, and A. Soghier. 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 93–104, 2009.

[49] E.K. Burke, R. Qu, and A. Soghier. 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'10)*, pages 136–151, 2010.

[50] E.K. Burke, R. Qu, and A. Soghier. Adaptive tie breaking and hybridisation in a graph-based hyper-heuristic for exam timetabling problems. In *Proceedings of the Nature Inspired Cooperative Strategies for Optimisation*, Studies in Computational Intelligence. Springer, 2011.

[51] E.K. Burke and E. Soubeiga. Scheduling nurses using a tabu-search hyperheuristic. In *Proceedings of the 1st Multidisciplinary Intl. Conf. on Scheduling: Theory and Applications (MISTA 2003)*, volume 1, pages 197–218, Nottingham, UK, August 13-16 2003.

[52] M. Caramia, P. Dell Olmo, and G.F. Italiano. Novel local-search-based approaches to university examination timetabling. *Informs Journal of Computing*, 20(1):86–99, 2008.

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

[54] M.W. Carter, G. Laporte, and S.Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of Operational Research Society*, 74:373–383, 1996.

[55] S. Casey and J.M. Thompson. Grasping the examination scheduling problem. In *Practice and Theory of Automated Timetabling: Selected papers from the 4th International Conference .Lecture Notes in Computer Science*, volume 2740, pages 232–244, 2003.

[56] V. Cerny. Thermo dynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[57] K. Chakhlevitch and P.I. Cowling. Choosing the fittest subset of low level heuristics in a hyperheuristic framework. in lncs ,. In *Evolutionary Computation in Combinatorial Optimization, 5th European Conference (EvoCOP 05)*, volume 3448 of *Lecture Notes in Computer Science*, pages 23–33, 2005.

[58] A. J. Cole. The preparation of examination timetables using a small-store computer. *Computer Journal*, 7:117–121, 1964.

[59] A.W. Colijn and C. Layfield. Conflict reduction in examination schedules. In E. Burke and P. Ross, editors, *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT'95)*, pages 297–307, 1995.

[60] R. Colome and D. Serra. Consumer choice in competitive location models: Formulations and heuristics. Technical report, Depart-

ment of Economics and Management, Universitat Pompeu Fabra, Barcelona, Spain, 1998.

[61] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

[62] P. Cowling and K. Chakhlevitch. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC 03)*, pages 1214–1221, 2003.

[63] P. Cowling, G. Kendall, and N. M. Hussin. A survey and case study of practical examination timetabling problems. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 02)*, pages 258–261, 2002.

[64] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In E. Burke and W. Erben, editors, *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling (PATAT 2000)*, pages 176–190. Springer LNCS, 2001.

[65] P. Cowling, G. Kendall, and E. Soubeiga. A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the 4th Metaheuristic International Conference*, pages 127–131, 2001.

[66] P. Cowling, G. Kendall, and E. Soubeiga. Hyperheuristics: A robust optimisation method applied to nurse scheduling. In *Proceedings of the VII Parallel Problem Solving From Nature (PPSN VlI)*, volume 2439 of *LNCS*, pages 7–11. Springer, 2002.

[67] P.I. Cowling, G. Kendall, and L. Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem.

In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 02)*, pages 1185–1190, 2002.

[68] A. Cuesta-Canada, L. Garrido, and H. Terashima-Marin. Building hyper-heuristics through ant colony optimization for the 2d bin packing problem. In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 05)*, volume 3684 of *Lecture Notes in Computer Science*, pages 654–660, 2005.

[69] P. David. A constraint-based approach for examination timetabling using local repair technique. In E.K. Burke and M.W. Carter, editors, *Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference*, volume 1408 of *Lecture Notes in Computer Science*, pages 169–186, 1998.

[70] G. De Smet. Examination track, practice and theory of automated timetabling. In *Examination track, Practice and Theory of Automated Timetabling (PATAT08), Montreal, 19-22, August*, 2008.

[71] D. De Werra. An introduction to timetabling problem. *European Journal of Operational Research*, 19(2):151–162, 1985.

[72] H. Delmaire, J.A. Diaz, E. Fernandez, and Ortega M. Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *INFOR*, 37:94–225, 1999.

[73] S. Desroches, G. Laporte, and J.M. Rosseau. Horex: A computer program for the construction of examination schedules. *INFOR*, 16:294–298, 1978.

[74] L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In E.K. Burke and W. Erben, editors, *Selected*

papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT'00), volume 2079 of *Lecture Notes in Computer Science*, pages 104–117, 2001.

[75] C. Dimopoulos and A.M.S Zalzala. Investigating the use of genetic programming for a classic one-machine scheduling problem. *Advances in Engineering Software*, 32(6)(@INPROCEEDINGSColijn1995, author = Colijn, A.W. and Layfield, C., title = Conflict reduction in examination schedules, booktitle = Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT'95), year = 1995, editor = Burke, E. and Ross, P., pages = 297-307, owner = Afaf, timestamp = 2011.04.17):489–498, 2001.

[76] M. Dorigo and L. Gambardella. Ant colonies for the traveling salesman problem. *Biosystems*, 43:73–81, 1997.

[77] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. on Systems, Man, and Cybernetics Part B*, 26:29–41, 1996.

[78] U. Dorndorf and E. Pesch. Evolution based learning in a job shop scheduling environment. *Computers and Operations Research*, 22:25–40, 1995.

[79] K. Dowsland, E. Soubeiga, and E.K. Burke. A simulated annealing hyperheuristic for determining shipper sizes. *European Journal of Operational Research*, Vol 179, issue 3:pp 759–774, 2007.

[80] K. Dowsland and J. Thompson. Ant colony optimization for the examination scheduling problem. *Journal of the Operational Research Society*, 56(4):426–438, 2005.

[81] G. Dueck. New optimisation heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104:86–92, 1993.

[82] T.A. Duong and K.H. Lam. Combining constraint programming and simulated annealing on university exam timetabling. In *Proceedings of the 2nd International Conference in Computer Sciences, Research, Innovation & Vision for the Future(RIVF2004)*, pages 205–210, 2004.

[83] M. Eley. Ant algorithms for the exam timetabling problem. In *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT04)*, pages 364–382, 2006.

[84] W. Erben. A grouping genetic algorithm for graph colouring and exam timetabling. In *The 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT00)*, volume 2079 of *Lecture Notes in Computer Notes*, pages 132–156. Springer, 2001.

[85] T. Feo and M. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[86] T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[87] H. Fisher and G.L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. In *Factory Scheduling Conference, Carnegie Institue of Technology*, pages 10–12, 1961.

[88] H. Fisher and G.L. Thompson. Probabilistic learning combinations of local job shop scheduling rules. *Industrial Scheduling*, pages 225–251, 1963.

[89] E. Foxley and K. Lockyer. The construction of examination timetables by computer. *The Computer Journal*, 11:264–268, 1968.

[90] A.S. Fukunaga. Evolving local search heuristics for sat using genetic programming. In K. Deb, R. Poli, W. Banzhaf, H.G. Beyer, E.K. Burke, P. Darwen, D. Dasgupta, D. Floreano, J. Foster, M. Harman, O. Holland, P.L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, and A. Tyrrell, editors, *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 04)*, volume 3103 of *Lecture Notes in Computer Science*, pages 483–494. Springer, 2004.

[91] P. Garrido and M.C. Riff. Collaboration between hyperheuristics to solve strippacking problems. In *Proceedings of the 12th International Fuzzy Systems Association World Congress*, volume 4529 of *Lecture Notes of Computer Science*, page 698707. Springer, 2007.

[92] P. Garrido and M.C. Riff. An evolutionary hyperheuristic to solve strip-packing problems. In *Proceedings of the Intelligent Data Engineering and Automated Learning (IDEAL 07)*, volume 4881 of *Lecture Notes in Computer Science*, page 406415. Springer, 2007.

[93] A. Gendreau, L. Salvail, and P. Soriano. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, 41:385–403, 1993.

[94] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549, 1986.

[95] F. Glover. Tabu search part i. *ORSA Journal on Computing 1*, 3:190–206, 1989.

[96] F. Glover. Tabu search part ii. *ORSA Journal on Computing 2*, 1:4–32, 1990.

[97] F. Glover and M. Laguna. *Modern Heuristics Techniques for Combinatorial Problems*, chapter Tabu Search, pages 70–141. Blackwell Scientific Publishing, 1993.

[98] F. Glover and M. Laguna. Tabu search. *Kluwer,Dordecht*, 1997.

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

[100] P. Gogos, C.and Alefragis and E. Housos. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, page to appear, 2010.

[101] D. Goldberg. *Genetic Algorithms in Search*, chapter Optimization and Machine Learning. Kluwer Academic Publishers, Boston, MA, 1989.

[102] L. Han and G. Kendall. Guided operators for a hyper-heuristic genetic algorithm. In T. D. Gedeon and L. C. C. Fung, editors, *Proceedings of AI-2003: Advances in Artificial Intelligence. The 16th Australian Conference on Artificial Intelligence (AI 03)*, pages 807–820, 2003.

[103] L. Han and G. Kendall. Investigation of a tabu assisted hyper-heuristic genetic algorithm. In *Proceedings of Congress on Evolutionary Computation (CEC 03)*, volume 3, pages 2230–2237, 2003.

[104] L. Han, G. Kendall, and P. Cowling. An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 02)*, pages 267–271, 2002.

[105] P. Hansen and N. Mladenovic. An introduction to variable neighborhood search. In S. Voss, I.H. Osman, and C. Roucairol, editors, *Metaheuristics, advances and trends in local search paradigms for optimization*, pages 433–458. Kluwer, 1999.

[106] J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.

[107] A. Hertz. Tabu search for large scale timetabling problems. *European Journal of Operations Research*, 54:39–47, 1991.

[108] N.B. Ho and J.C. Tay. Evolving dispatching rules for solving the flexible jobshop problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 05)*, pages 2848–2855, 2005.

[109] J.H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992.

[110] D. Jakobovic, L. Jelenkovic, and L. Budin. Genetic programming heuristics for multiple machine scheduling. In *Proceedings of the European Conference on Genetic Programming (EUROGP 07)*, volume 4445 of *Lecture Notes in Computer Science*, pages 321–330, 2007.

[111] D. Johnson. Timetabling university examinations. *Journal of Operational Research Society*, 41:3947, 1990.

[112] D.E. Joslin and D.P. Clements. Squeaky wheel optimization. *Journal of Artificial Intelligence Research*, 10:353–373, 1999.

[113] R.E. Keller and R. Poli. Linear genetic programming of parsimonious metaheuristics. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 07)*, pages 4508–4515, 2007.

[114] G. Kendall and N. Hussin. An investigation of a tabu search based hyper-heuristic for examination timetabling. In G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, editors, *Selected Papers from MISTA 2005*, pages 309–328. Springer, 2005.

[115] G. Kendall and N.M. Hussin. A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology. In E.K. Burke and M. Trick, editors, *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, volume 3616 of *Lecture Notes in Computer Science*, pages 199–218. Springer, 2005.

[116] C. Kenyon. Best-fitt bin-packing with random order. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 359–364, 1996.

[117] S. Kirkpatrick, C.D. Gelatt, and M.P Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[118] N. Krasnogor. Self generating metaheuristics in bioinformatics: The proteins structure comparison case. *Genetic Programming and Evolvable Machines*, 5(2):181–201, 2004.

[119] N. Krasnogor and S. Gustafson. A study on the use of "self-generation" in memetic algorithms. *Natural Computing*, 3(1):53–76, 2004.

[120] R. Kumar, A.H. Joshi, K.K. Banka, and P.I. Rockett. Evolution of hyperheuristics for the biobjective 0/1 knapsack problem by multiobjective genetic programming. In *Proceedings of the 10th ACM conference on Genetic and evolutionary computation (GECCO 08)*, pages 1227–1234, 2008.

[121] M. Laguna and Gonzalez-Velarde. A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent manufacturing*, 2:253–260, 1991.

[122] G. Laporte and S. Desroches. Examination timetabling by computer. *Computers & Operations Research*, 11:361372, 1984.

[123] B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and S. Abdullah. An extended great deluge approach to the examination timetabling problem. In *Proceedings of the 4th Multidisciplinary International Scheduling: Theory and Applications 2009 (MISTA 2009), pp. 424-434, 10-12 August , Dublin, Ireland*, 2009.

[124] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, and L. Parkes, A.J.and Di Gaspero. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1), 2010.

[125] N.K. Mehta. The application of a graph coloring method to an examination scheduling problem. *Interfaces*, 11:57–64, 1981.

[126] N. Merlot, L.and Boland, B. Hughes, and P. Stuckey. A hybrid algorithm for the examination timetabling problem. In E.K. Burke and P. Causmaecker, editors, *Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference (PATAT02)*, volume 2740 of *Lecture Notes in Computer Science*, pages 207–231, 2003.

[127] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech Concurrent Computation Program 826, California Institute of Technology, 1989.

[128] P. Moscato and M.G. Norman. A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. *Parallel Computing and Transputer Applications*, pages 177–186, 1992.

[129] T. Muller. Itc 2007 solver description: A hybrid approach. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.

[130] M. Oltean. Evolving evolutionary algorithms using linear genetic programming. *Evolutionary Computation*, 13(3):387–410, 2005.

[131] I.H. Osman and J.P. Kelly, editors. *Metaheuristics: Theory and Applications*. Kluwer, Dordrecht, 1996.

[132] E. Ozcan, Y. Bykov, M. Birben, and E.K. Burke. Examination timetabling using late acceptance hyper-heuristics. *Evolutionary Computation, 2009(CEC '09)*, pages 997 – 1004, 2009.

[133] G.L. Pappa and A.A. Freitas. Automatically evolving rule induction algorithms tailored to the prediction of postsynaptic activity in proteins. *Intelligent Data Analysis*, 13(2):243–259, 2009.

[134] L. Paquete and T. Stutzle. Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling(PATAT'02)*, pages 21–23. Springer-Verlag, 2002.

[135] N. Pillay. A developmental approach to the examination timetabling problem. In *Practice and Theory of Automated Timetabling (PATAT 2008)*, pages 19–22, August 2008.

[136] N. Pillay. 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'10)*, pages 336–346, 2010.

[137] N. Pillay. A study into the use of hyper-heuristics to solve the school timetabling problem. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '10)*, 2010.

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

[139] N. Pillay and W. Banzhaf. An informed genetic algorithm for the examination timetabling problem. *Applied Soft Computing*, 10(2):457–467, 2010.

[140] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403–2435, 2007.

[141] R. Poli, J.R. Woodward, and E.K. Burke. A histogram-matching approach to the evolution of bin-packing strategies. In *Proceedings of the Congress on Evolutionary Computation (CEC 2007)*, pages 3500–3507, 2007.

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

[143] R. Qu, E.K. Burke, and B. McCollum. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring

problems. *European Journal of Operational Research*, 198(2):392–404, 2009.

[144] R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee. A survey of search methodologies and automated approaches for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.

[145] P. Rattadilok, A. Gaw, and R. Kwan. Distributed choice function hyper-heuristics for timetabling and scheduling. In E.K. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture notes in Computer Science*, pages 51–67. Springer, 2005.

[146] P. Ross. Hyper-heuristics. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 529–556. Kluwer, Boston, 2005.

[147] P. Ross, E. Hart, and D. Corne. Some observations about ga-based exam timetabling. In *Selected papers from the Second International Conference on Practice and Theory of Automated Timetabling (PATAT 97)*, pages 115–129, 1998.

[148] P. Ross, J.G. Marin-Blazquez, S. Schulenburg, and E. Hart. Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*, pages 1295–1306, 2003.

[149] P. Ross, S. Schulenburg, J.G. Marin-Blazquez, and E. Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'02)*, pages 942–948, 2002.

[150] K. Sheibani. An evolutionary approach for the examination timetabling problems. In E. Burke and P. Causmaecker, editors, *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 02)*, pages 387–396, 2002.

[151] K. Socha, J. Knowles, and M. Sampels. A max- min ant system for the university timetabling problem. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 - Third International Workshop on Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 1–13. Springer, Berlin, Germany, 2002.

[152] R.H. Storer, S.D. Wu, and R. Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10):1495–1509, 1992.

[153] R.H. Storer, S.D. Wu, and R. Vaccari. Problem and heuristic space search strategies for job shop scheduling. *ORSA Journal on Computing*, 7(4):453–467, 1995.

[154] J. Tavares, P. Machado, A. Cardoso, F.B. Pereira, and E. Costa. On the evolution of evolutionary algorithms. In M. Keijzer, U.M O'Reilly, S.M. Lucas, E. Costa, and T. Soule, editors, *Proceedings of the European Conference on Genetic Programming (EUROGP 04)*, volume 3003 of *Lecture Notes in Computer Science*, pages 389–398, 2004.

[155] J.C. Tay and N.B. Ho. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering*, 54(3):453–473, 2008.

[156] H. Terashima-Marin, E.J. Flores-Alvarez, and P. Ross. Hyper-heuristics and classifier systems for solving 2d-regular cutting stock

problems. In *Proceeedings of the ACM Genetic and Evolutionary Computation Conference (GECCO 05)*, pages 637–643, 2005.

[157] H. Terashima-Marin, C.J.F. Zarate, P. Ross, and M. Valenzuela-Rendon. A ga-based method to produce generalized hyper-heuristics for the 2d-regular cutting stock problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 06)*, pages 591–598, 2006.

[158] J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Computers and Operations Research*, 25(7):637–648, 1998.

[159] J. M. Thompson and K. A.. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, 63:105128, 1996.

[160] J.A. Vazquez-Rodriguez, S. Petrovic, and A. Salhi. A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup times and uniform machines. In P. Baptiste, G. Kendall, A. Munier, and F. Sourd, editors, *Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 07)*, 2007.

[161] D.J.A. Welsh and M.B. Powell. The upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 11:41–47, 1967.

[162] B. White, G. Xie and S. Zonjic. Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, 153(16):80–91, 2004.

[163] G.M. White and B.S. Xie. Examination timetables and tabu search with longer-term memory. In E.K. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, Lecture Notes in Computer Science 2079, pages 85–103. Springer, Berlin, 2001.

[164] D. Whitley and J. P. Watson. Complexity and no free lunch. In E.K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 317–339. Kluwer, Boston, 2005.

[165] D. Whitley and J.P. Watson. Complexity theory and the no free lunch theorem. In E.K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Kluwer, Boston, 2005.

[166] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[167] T. Wong, P. Cote, and P. Gely. Final exam timetabling: a practical approach. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*, pages 726–731, 2002.

[168] D.C. Wood. A system for computing university examination timetables. *The Computer Journal*, 11:4147, 1968.

[169] D.C. Wood. A technique for colouring a graph applicable to large scale timetabling problems. *The Computer Journal*, 12:317–319, 1969.