HYBRID META-HEURISTIC ALGORITHMS FOR UNIVERSITY COURSE TIMETABLING PROBLEMS

KHALID SHAKER JASIM

THESIS SUBMITTED IN FULFILMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FAKULTI TEKNOLOGI DAN SAINS MAKLUMAT
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2011

ALGORITMA HIBRID META-HEURISTIK UNTUK MASALAH PENJADUALAN KURSUS UNIVERSITI

KHALID SHAKER JASIM

TESIS YANG DIKEMUKAKAN UNTUK MEMPEROLEH IJAZAH
DOKTOR FALSAFAH

FAKULTI TEKNOLOGI DAN SAINS MAKLUMAT
UNIVERSITI KEBANGSAAN MALAYSIA
BANGI

2011

DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged.

1 July 2011

KHALID SHAKER JASIM
P39452

ACKNOWLEDGMENT

All the praises be to the mighty Allah, the Merciful and the Beneficent for the strength and blessing in the completion of this study.

Indeed there are many wonderful people who have contributed significantly throughout the whole course of my study up to the completion of this thesis. I owe a great deal to them.

First and foremost, I wish to express my most sincere acknowledgment to my supervisor: Assoc. Prof. Dr. Salwani Abdullah for her valuable guidance, generosity and freedom throughout the entire research and thesis writing. Sincere appreciation goes to Professor Dr. Masri Ayob for her constructive comment. Many thanks for Dr. Paull McMullan for the encouragement, thoughtful comments and helpful discussion.

To my dearest mother and father, thank you for bringing me up to who I am today. My success symbolizes and reflects the support and love from both of you. My deepest appreciation goes to my wife and children for their love, patience and understanding.

I am very grateful to UKM and all the staff and members of the School of Computer Science for the help, and friends who have assisted me whenever I need them.

ABSTRACT

University course timetabling problem represents an NP-hard optimisation problem. It deals with an assignment of events and resources to timeslots and rooms while satisfying various constraints. The key problem associated with course timetabling problems is to find a high quality timetable. However, it is typically a difficult and time consuming task due to the size of the problem and satisfying all hard constrains while minimising the soft constrains violations as far as possible. The aim of the research represented in this thesis is to provide effective approaches for finding good quality solutions for university course timetabling. This has been achieved via a number of meta-heuristic approaches. The research first highlights a hybridisation approach with the aim integrating of the mechanism of Great Deluge Algorithm with the mechanism of Tabu Search Algorithm using number of neighbourhood structures. Next, the research investigates a different neighbourhood structure i.e. a Kempe Chain neighbourhood structure that is employed within the Great Deluge algorithm with an assumption that Kempe Chain neighbourhood structures can help in obtaining better solutions during the search process. The selection of the neighbourhood structures is later been controlled using a Round Robin algorithm, where each neighbourhood structure is given a time slice to be employed. This mechanism is embedded in a novel Dual Simulated Annealing algorithm. All of the algorithms discussed above are single-based solution approaches. The final algorithm investigated in this thesis is the ability of the hybridisation of population-based approach (Bacteria Swarm Optimisation Algorithm). This uses small neighbourhood structures with a Deferential Evaluation Algorithm in order to explore the search space while looking for better solutions. The performance of each algorithm is tested on the standard enrolment-based course timetabling problems and the curriculum-based international timetabling competition (ITC2007) datasets. Computational results show that in most of the cases, the presented approaches significantly outperform other available techniques on the established benchmark course timetabling problems.

ABSTRAK

Masalah penjadualan waktu kursus universiti merupakan masalah pengoptimuman NP-Sukar. Masalah ini melibatkan proses pengumpukan kursus kepada waktu kursus dan bilik kuliah dengan mempertimbangkan beberapa kekangan keras. Masalah utama dengan penjadualan waktu adalah untuk mencari jadual yang berkualiti tinggi. Walau bagaimanapun, ia merupakan suatu tugas yang sukar dan memerlukan masa yang lama untuk diselesaikan disebabkan oleh saiz masalah, cubaan mematuhi kekangan wajib dan meminimikan perlanggaran kekangan pilihan. Matlamat penyelidikan ini adalah untuk menyediakan pendekatan yang efektif untuk menyelesaikan masalah penjadualan jadual waktu kursus universiti. Matlamat ini telah dicapai melalui beberapa pendekatan meta-heuristik. Penyelidikan ini dimulakan dengan pembangunan pendekatan hibrid iaitu pendekatan *Great Deluge* diintegrasikan dengan algoritma *Tabu Search* menggunakan struktur kejiiranan yang kecil. Ini dikuti dengan penyelidikan sturuktur kejiranan *Kempe Chain* yang diguna pakai bersama dengan pendekatan *Great Deluge* dengan andaian struktur kejiranan *Kempe Chain* berupaya membantu mendapatkan penyelesaian yang lebih baik semasa proses carian. Pemilihan struktur kejiranan kemudiannya dikawal oleh algoritma *Round Robin*. Setiap struktur kejiranan diumpukkan tempoh masa untuk digunakan. Mekanisma ini diimplementasikan bersama dengan algoritma *Dual Simulated Annealing*. Kesemua algoritma yang dibincangkan di atas adalah pendekatan berasaskan penyelesaian-tunggal. Sumbangan terakhir adalah untuk menyelidik keupayaan menghibridisasikan pendekatan berasaskan populasi iaitu *Bacteria Swarm Optimization* yang menggunakan struktur kejiranan yang kecil dengan algoritma *Deferential Evaluation* dalam meneroka ruang carian bagi mendapatkan penyelesaian yang lebih baik. Prestasi setiap algoritma diuji pada masalah piawai penjadualan waktu kursus berasaskan enrolmen dan set data pertandingan penjadualan antarabangsa berasaskan kurikulum (ITC2007). Dalam kebanyakan kes, keputusan ekperimen menunjukkan prestasi pendekatan yang dicadangkan adalah lebih baik berbanding pendekatan lain dalam masalah penjadualan waktu kursus.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

Timetabling problems are considered as a specific type of scheduling problem. It is concerned with the assignment of events to timeslots subject to constraints with the resultant solution constituting a timetable. It is known as a NP-hard problem (Schaerf, 1999). One example of timetabling problems is the University Course Timetabling Problem (UCTP) which is a significant problem in higher educational institutions.

UCTP which is also sometimes known as class/teacher timetabling, refer to a set of courses that need to be scheduled into a given number of rooms and timeslots within a week, and at the same time, students and teachers are assigned to courses so that the meetings can take place. Carter and Laporte (1998) defined course timetabling as:

"a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers)are assigned to classrooms and times"

There are two goals involved in solving course timetabling problems. Firstly, to produce feasible timetables while taking into consideration satisfying a number of criteria called hard constraints. Secondly, to produce good quality timetables by reducing the

violations on a number of undesirable criteria called soft constraints (for more details about the hard and soft constraints, see Chapter III). A feasible timetable is one in which all the courses can have all of the resources (might be rooms with particular equipment, students, and lecturers) that they require at a particular timeslot (that has been allocated to them), of course. A good quality timetable is the one that conforms well to a number of soft constrains that are set by the user.

Traditionally, the task of scheduling the courses in the university is carried out manually based on trial and hit, which normally take days or weeks to find a clash free timetable. However, sometimes a better timetable is not guaranteed. Thus, researchers in the area of Artificial Intelligence and Operation Research have paid serious attention to provide automated support for human timetables.

In the last few years, researchers have attempted to investigate a lot of methods to solve university course timetabling problem (see Schaerf 1999; Burke and Petrovic 2002; McCollum 2007; Lewis 2008; McCollum et al. 2009). Early research concentrated on sequential heuristics and later moved to meta-heuristic approaches due to the ability of these approaches to generate solutions which are better than those generated from sequential heuristics alone (Schaerf 1999, Burke and Petrovic 2002)

Usually, in university course timetabling, an initial solution will be constructed using one or more appropriate heuristics (least saturation degree, large degree or greedy heuristic, etc) and then the improvement is carried out using meta-heuristics. However, the performance of meta-heuristic approaches may differ from instance to other instances, which might depend on the parameter tuning process, the neighbourhood structures and the search algorithm itself (Burke et al. 2004). Thus, the research work presented in this thesis is focused on controlling the selection of the neighbourhood structures and enhancing the search algorithm by hybridising between two or more meta-heuristic approaches to better explore the search space while finding better solutions.

A brief observation of the recent timetabling literature shows that most of the hybridisation approaches have been applied successfully in the past to a number of difficult combinatorial optimisation problems, particularly on scheduling problems. This motivates the investigation of a hybridisation approach between meta-heuristic approaches in order to have a benefit of the exploration and exploitation mechanism during the search process. This thesis is derived from the interest to develop effective automated course timetabling hybridisation techniques. These hybrid approaches are expected to drive towards achieving better solutions and in a more effective way.

1.2 PROBLEM STATEMENT

The university course timetabling problem deals in building up a weekly course timetable that share the timeslot and the room for each course over a given semester or year. It involves the arrangement of courses, students, teachers and rooms at a specific number of timeslots, respecting a certain constraints. Constraints in course timetabling problem can be classified as hard and soft constraints. Hard constraints must be satisfied by a solution of the problem (to generate a feasible timetable), whereas soft constraints are desirable to be satisfied in order to obtain a good timetable (Socha et al. 2002). Examples of such hard constraints are: teachers cannot give more than one lecture at one time, the student cannot be assigned to more than one course at the same time, and not more than one course is allowed to be assigned to a timeslot in each room. Examples of such hard constraints are: student has more than two consecutive courses, and student has a single course on a day. Note that, details description on hard and soft constraints can be found in Chapter III.

Course timetabling problem is one of major administrative activities in higher education institutes (Burke and Petrovic 2002). These institutes spend a lot of money and efforts just to get feasible or good timetables in a reasonable time. Most of the time, the output is not as planned or desired. The teachers prefer to give a lectures at the beginning of the planning period, whereas the students' interest to have lectures at different period. Thus, to create this kind of timetable manually that tries to satisfy all preferences by teachers and student is normally a time consuming task and need high effort from human.

Hence, the only solution is to automate a generation of timetables that normally take a shorter time compared to manual. This can be considered as cost cutting effort as well.

In the universities, using good timetables became more significant in recent years (Bykov 2003). But the lack of proper resources (e.g. number of rooms, their capacities and availability of lecturers) create problem to the university timetabling process which requires expensive human and computer resources for solving it. Timetables that are not properly generated will create a negative impact to the faculties such as a class room clashes, subject conflicts etc. Thus, efficient techniques should be available to generate a high quality and clash free timetables.

However, satisfying 100% of the soft constraints in order to generate a high quality timetable is a very difficult task or maybe it is impossible (Qu et al. 2009). Since a large number of variants of the problem that differs from each other, which is based on the categories of universities and their constraints, it is very difficult to formulate a general problem that serve the requirements of all universities (Lewis 2008). A general technique should include all possible aspects which can easily be simplified as the requirements of the users. The number and type of constraints are different from one university to others. This leads to a corresponding difference in objective functions, which provide numerical measures of violation of these constraints.

Due to the complexity of the university course timetabling problems, and the limitations of proposed methods in the literature, thus, solving university course timetabling problems that are based on meta-heuristic algorithms still need more investigations in term of the parameters used, selection of the neighbourhood structures and the search algorithm itself.

Some of the research questions that can be pointed out are:

i. Does the hybridisation of the meta-heuristic methods can create a balance between exploration and exploitation during the search process?

ii. Do the meta-heuristic methods that are employed on different search regions contribute to a better search process?

iii. Do the small neighbourhood structures can be efficient in terms of the time require for a search algorithm?

iv. How meta-heuristic methods can utilise simple neighbourhood structures?

v. What is the effect of using a large size neighbourhood structure compared to small neighbourhood structures in increasing the probability of finding high quality solutions?

vi. Does a large size neighbourhood structure helps to find feasible path between the search space regions?

vii. Does the neighbourhood structure selection strategy helps to increase the performance of search algorithm?

viii. How the convergence speed is affected by the population based method operators?

1.3 RESEARCH OBJECTIVES

The overall aim of the work in this thesis is to assess the ability of hybridisation meta-heuristic approaches in solving university course timetabling problem. This research aims to propose a hybridisation approaches to generate good quality timetables (solutions) for two university course timetabling problems (see Chapter III). In order to accomplish this aim, several objectives are outlined as follows:

1. To propose a great deluge and tabu search hybrid technique with simple neighbourhood structures.

2. To propose the hybridization of large size of neighbourhood structures with great deluge algorithm.

3. To propose a new method called Dual sequence simulated annealing algorithm with round robin algorithm in handling university course timetabling.

4. To propose a hybrid population-based approach i.e. bacteria swarm optimisation algorithm and differential evolution algorithm to increase the ability of exploration and exploitation process.

Figure 1.1 shows directions for each objective i.e. which objective will answer which research questions.

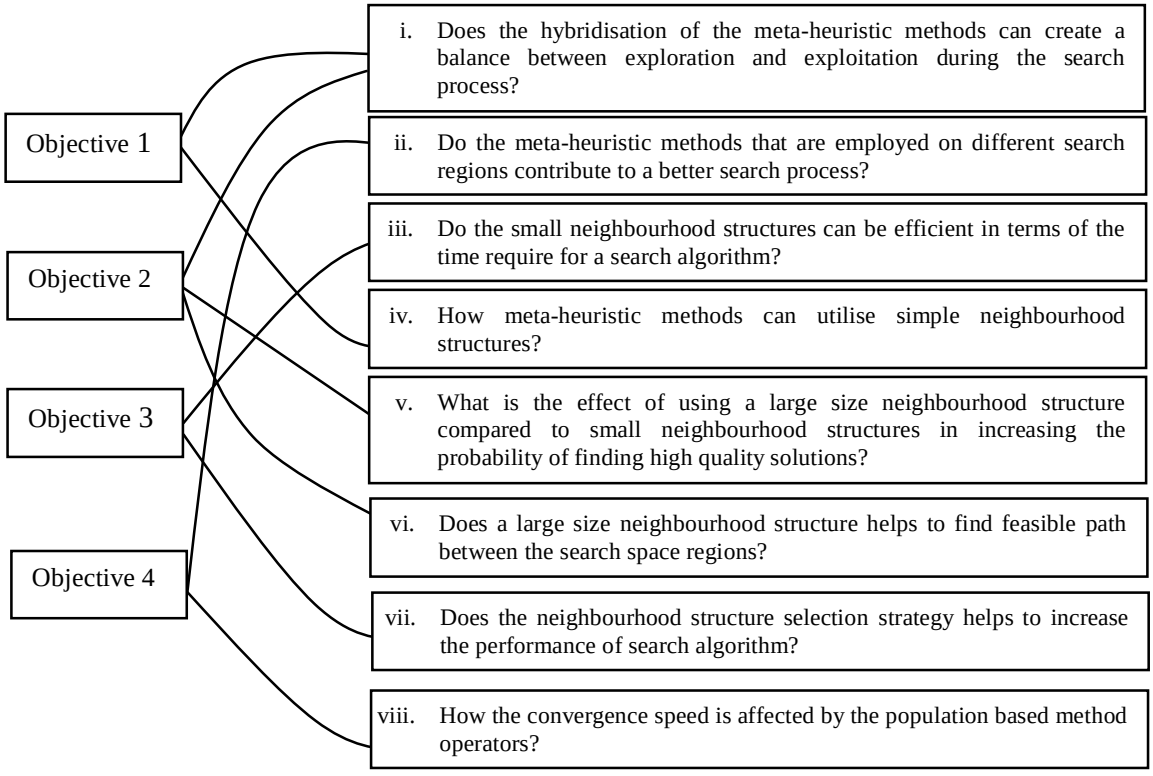

Figure 1.1 The objectives answer the research questions

1.4 RESEARCH SCOPE

The research focuses on the hybridisation of meta-heuristic algorithms in solving university course timetabling problems. The proposed approaches in this thesis are applied on standard benchmark datasets. These datasets are varied in term of size, constraints involved and conflict density. The algorithms are tested on two sets of standard benchmark datasets classified as enrolment-based course timetabling problems (11 datasets), and curriculum-based course timetabling problems ITC2007-Track 3 (UD1 (32 datasets), UD2 (32 datasets)). The experimental results (with respect to the quality of solutions) are compared against available approaches in the literature that tackled the same problems. The performance of the algorithms is evaluated based on the results obtained.

1.5 OVERVIEW OF THE THESIS

This thesis consists of nine chapters. This chapter presents the background and motivation, problem statement, research objectives and scope. The remainder of this thesis is organised as follows:

Chapter II presents the literature review of the related studies in university course timetabling problems. It introduces the timetabling problem in general, and then concentrates upon reviews and analyses the current published researches on this problem.

Chapter III demonstrates the methodology of the research. Three phases (i.e. preprocessing phase, construction phase and improvement phase) of the methodology are described. Preprocessing phase concentrates on transferring the original data into the related data structures (matrixes). Construction phase involves in generating initial solutions for each dataset tested in this work. Later the quality of these solutions will be enhanced at the improvement phase. In addition, this chapter also describes the specification of the datasets, hard and soft constraints involved, and the objective function that is employed to measure the quality of the obtained solutions.

Chapter IV investigates the incorporation of the great deluge and tabu search algorithms to solve the university course timetabling problem. It involves two experiments (preliminary and extended). The aim of the preliminary experiment is to measure the effectiveness of the simple neighbourhood structures used. During the search, only the most frequently used neighbourhood structures will be used in the extended experiment with a relaxed stop condition. Tabu search is used to prevent cycling of the some neighbourhood structures. A used/unused strategy is used to control the selection of neighbourhood structures.

In Chapter V, a different size of neighbourhood structures i.e. a kempe chain neighbourhood structure is applied within the same algorithm as in Chapter IV. This is due to the experimental analysis in Chapter IV that different size or complexity of the problem might need different neighbourhood structures in order to help the search algorithm to explore the search space. This helps in improving the efficiency of the algorithm in generating better solutions. The round robin algorithm is utilised to control the selection of the neighbourhood structures. This effectively helps the algorithm to diversify the search process in getting better solutions.

Chapter VI presents a novel dual sequence simulated annealing approach to solve the university course timetabling problem. The approach consists of two schemas that are used to avoid the stagnation state. In the first schema, the worse solution is accepted if there is no improvement after a certain number of iterations, referred as a local counter. The second schema uses a counter, referred as a global non improvement counter to start with a new sequence (with a new initial solution). The round robin algorithm is also employed here to control the selection of the neighbourhood structures.

Chapter VII describes a simulation of a bacteria swarm optimisation algorithm and applied to university course timetabling problems. The proposed algorithm simulates the movement of bacteria through searching for nutrient in the search space. The search space is divided into three regions based on the quality of solutions. These regions is classified as *risk* region, *null* region and *rich* region. The solutions in *risk* region and *null*

region have to move toward *rich* region, and solutions in the *rich* region have to move toward the best solution in the same region. Differential evolution algorithm is employed within the bacteria swarm optimisation to direct the search toward the potential regions in the search space.

Chapter VIII presents an analysis and evaluation of the results obtained from four proposed improvement algorithms (as presented in Chapters IV~VII) to solve university course timetabling problems.

Finally, the overall conclusions of the work presented in this thesis and research directions for future work in this area are presented in Chapter IX.

CHAPTER II

OVERVIEW OF ALGORITHMIC APPROACHES TO COURSE TIMETABLING PROBLEMS

2.1 INTRODUCTION

This chapter provides an overview of the different approaches presented in the literature to solve the university course timetabling problems. It discusses the major purposes of these techniques and gives an indication of further research guidelines in this area. Due to the vast number of published work in this area, this chapter only focuses on the most significant works in the literature.

This chapter is organised as follows: Section 2.2 presents the general definition of university course timetabling problem. Section 2.3 discusses a graph coloring for university timetabling problems. Section 2.4 presents most popular techniques applied to university course timetabling problems. Section 2.5 presents a summary of the chapter.

2.2 TIMETABLING PROBLEMS

Timetabling is one of the major issues faced by any university around the world that needs a lot of human and computer efforts to solve. The process of timetabling deals with finding suitable timeslots for a variety of functions with inadequate resources. This in-adequacy creates scheduling problems. Depending on the nature of the problem, the constraints can vary with various objectives.

Wren (1996) defined timetabling as:

"Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives."

Scheduling problems occur in many fields. One of them is called university timetabling problems. University timetabling problems are difficult tasks faced by educational institutions. These problems can be classified into two main categories i.e. course timetabling and examination timetabling (Schaerf 1999). These two categories share the same basic characteristics of the general timetabling problem but have different constraints that are usually divided into two classes i.e. hard and soft (Burke et al. 2004). In exam timetabling problems, a number of exams can be scheduled in the same room at the same timeslot (providing seating-capacity constraints are not exceeded), whilst in course timetabling problems, generally only allowed one course per room, per timeslot (Lewis 2008). The objective is to find a viable course schedule (feasible timetable) based on different constraints. The course scheduling and examination scheduling problems are common in many ways and dissimilar in other ways. One example for the common problem characteristics is that students cannot attend two classes or two exams simultaneously. Whereas, the examination scheduling problem may not have a fixed time period and normally is scheduled once per semester (or per year), but all course scheduling problems are for fixed time periods and normally based on weekly basis.

Fox and Sadeh-Koniecpol (1990) defined the scheduling concept as follows:

"*Scheduling selects among alternative plans, and assigns resources and times to each activity so that they obey the temporal restrictions of activities and the capacity limitations of a set of shared resources*."

The problem mainly occurs when trying to create a most appropriate timetable that balances the requirements of the management, lecturers and students. The problem is so critical that in some cases it is impossible to find even a single feasible

timetable. It is highly impossible to create a common model that can be used for all the universities, as every institute has their own specific constraints and objectives.

The later discussions only focus on course timetabling problems, which are considered as the domain in this research work.

2.3 UNIVERSITY COURSE TIMETABLING PROBLEMS

This thesis addresses university course timetabling problems (UCTT). Where, a set of courses are scheduled into a given number of rooms and timeslots across a period of time. This usually takes place at the beginning of a semester also the students and teachers will be assigned to courses so that the teaching delivery activities can take place (Schaerf 1999). During the process of timetabling, a number of problems might occur due to different hard and soft constraints. Basically a set of hard constraints need to be satisfied in order to produce a feasible solution, apart from satisfying the soft constraints as many as possible. Carter and Laporte (1998) defined course timetabling as:

"a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers) are assigned to classrooms and times"

Laporte and Desroches (1986) presented a problem of assigning students to course sections in a large engineering school. Carter and Laporte (1998) divided the course scheduling problem into five sub-problems such as course timetabling, student scheduling, class-teacher timetabling teacher assignment and classroom assignment. In addition, Lewis et al. (2007) have summarised the university course timetabling problem as the process of assigning lectures to a limited set of timeslots, while trying to satisfy some constraints.

The timetabling problem can be modeled using an undirected graph (De Werra 1985). University course timetabling problems can be viewed as a graph coloring model. Graph coloring is concerned with coloring the vertices of a given graph using a

given number of colors. The vertices represent the courses, the colors represent the timeslots and the edges represent the conflicts between courses Burke et al. (2004). Each vertex of a graph should be colored so that no two vertices that are connected by an edge are both assigned to the same color, and normally there are a limited number of available colors. A wide discussion on the graph coloring problem and its relationship to timetabling can be found in (De Werra 1996b, Burke and Ross 1996, De Werra 1997, and Burke et al. 2004).

This thesis deals with two sets of university course timetabling problems, namely, Enrolment-Based Course Timetabling Problems (EBCTT) and Curriculum-Based Timetabling Problems (CBCTT). The details of the specification of real-world course timetabling problems which are used as benchmark datasets are discussed in Chapter III along with the representation of hard and soft constraints and their corresponding objective functions.

2.4 TECHNIQUES APPLIED TO THE UNIVERSITY COURSE TIMETABLING PROBLEM

Various methods have been investigated to tackle university course timetabling problems. The interested readers can refer to the comprehensive survey of the problem in (Schaefer 1999, Burke and Petrovic 2002, Lewis 2008, and McCollum et al. 2010). For a gap between theory and practice in the area of university timetabling will be referred to McCollum (2006).

Carter and Laporte (1996) divided these methods into four categories such as sequential, meta-heuristics, constraint-based and cluster methods. A few years later, Petrovic and Burke (2004) included some other types such as case-based reasoning techniques, multi-criteria approaches and hyper-heuristic methods. On the other hand, Abdullah (2006) roughly classified the approaches applied on course timetabling problems into ten categories as follows: constraint-based methods, graph-based approaches, cluster-based methods, population-based approaches, meta-heuristic methods, multi-criteria approaches, hyper-heuristic/self adaptive approaches, case-based reasoning, knowledge-based and fuzzy-based approaches.

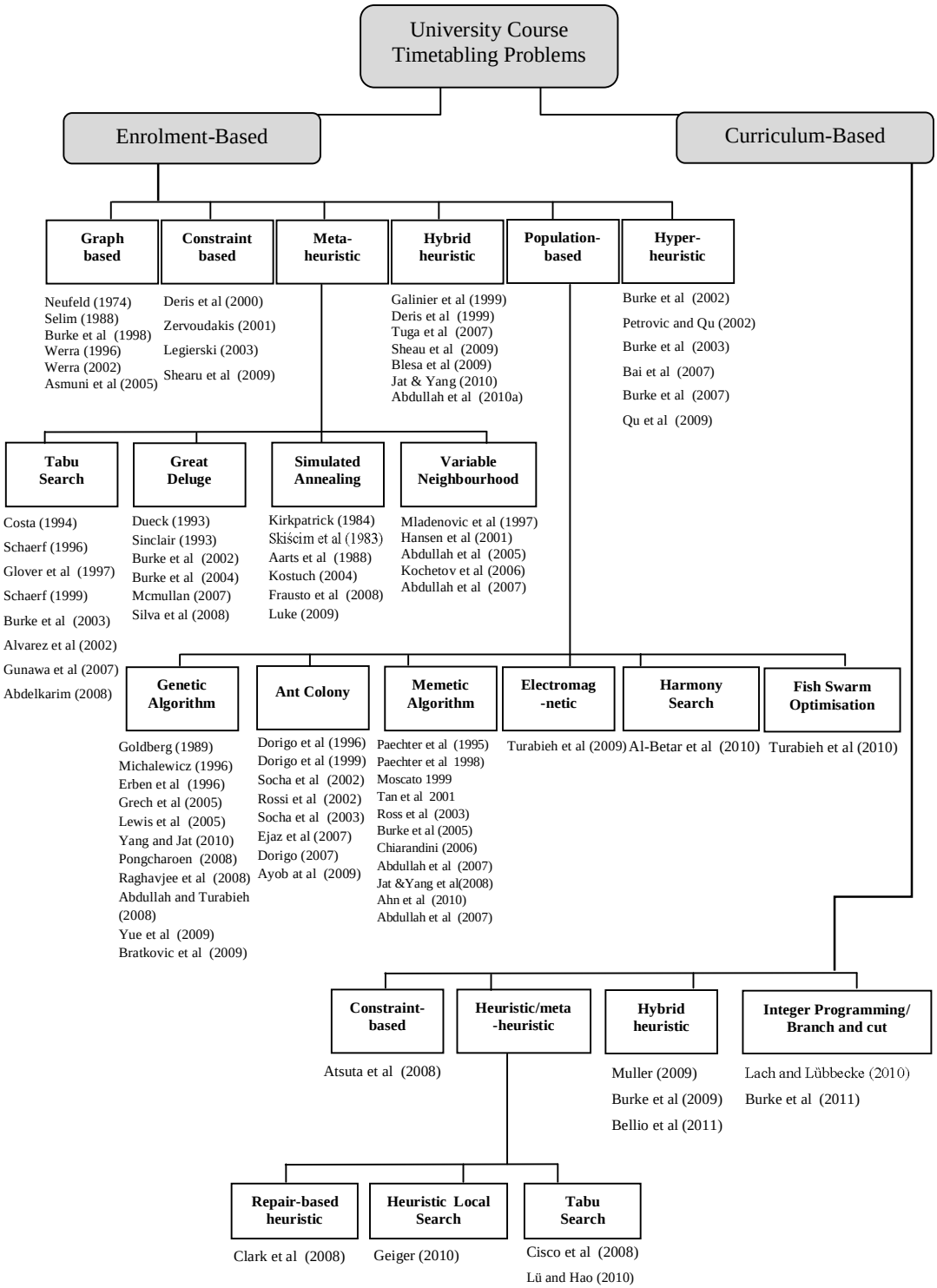

Figure 2.1 Summary of the employed approaches on course timetabling problems

As mentioned earlier, this chapter focuses on the Enrolment-Based Course Timetabling Problem (EBCTT) and the Curriculum-Based Timetabling Problem (CBCTT), which has been divided into six categories i.e. graph-based approaches, constraint-based approaches, meta-heuristic approaches, hybridisation meta-heuristic approaches, population-based approaches and hyper-heuristic algorithms. The category for EBCTT is divided into four i.e. constraint based, heuristic/meta-heuristic, hybridisation method, integer programming/branch and cut. Figure 2.1 summarises the available methods in the literature review that have been employed on both course timetabling problems.

2.4.1 Enrolment-Based Course Timetabling Problems

Plenty of approaches proposed in last three decades attempt to solve course timetabling problems and specially the enrolment-based course timetabling problem (full details about the problem can be found in Chapter III). In this section the approaches applied on this problem are discussed.

A. Graph-based Methods

As mentioned in Section 2.3, university course timetabling problems can be viewed as a graph coloring model where the vertices represent the courses, the colors represent the timeslots and the edges represent the conflicts between courses.

Neufeld and Tartar (1974) implemented a graph coloring method for a class-teacher timetabling problem. On the other hand, Selim (1988) employed a graph coloring methodology for the faculty timetable problem where the vertices were divided to reduce the chromatic number. The approach was tested on the real data from the Faculty of Science of the American University in Cairo.

De Werra (1996b) applied graph coloring models for course timetabling problems. This method has also been used by Asratian and De Werra (2002) to solve a class-teacher problem due to its ability to handle several disjoint groups of lectures.

Burke et al. (1998) investigated the effect of introducing a random element in the employment of graph coloring/timetabling heuristics by proposing two selection approaches: 1) tournament selection and 2) bias selection. The authors tested the proposed approaches over standard benchmark datasets. The results indicate that there are some improvements if a backtracking process is involved in the construction of feasible timetables.

Asmuni et al. (2005) applied a fuzzy-based algorithm to order courses based on graph coloring heuristics. Three heuristics (saturation degree, largest degree, and largest enrolment) have been employed. Experimental results show that this approach able to produce good quality solutions.

B. Constraint-based Methods

There are a few papers that discuss the use of Constraint-based methods in university course timetabling. In a constraint-based method, a problem is modeled as a set of variables with a finite domain. The method allocates values to variables that satisfy a number of constraints. Deris et al. (2000) formulated a timetable planning problem as a constraint-based reasoning technique which is implemented in an object oriented approach for colleges that involved 378 timeslots, 1673 subject sections and 10 rooms. They have introduced some constraint categories such as time, space and dispersion constraints. To facilitate the search for a solution, the timetabling problem is represented as a graph tree. In order to find the solution faster, variable orderings are introduced based on size (for example the size of the domain and the quantity of constraints of the variables). Verified results showed that the feasible and best solutions can be found in a reasonable time.

Zervoudakis and Stamatopoulus (2001) applied a constraint programming object-oriented model using ILOG SOLVER C++ library for the university course timetabling faced by the Department of Informatics and Telecommunications at the University of Athens. The problem involved 68 lectures that need to be scheduled in five days of nine teaching timeslots. A variety of search methods (for example depth

first search) and variable ordering heuristics were used in searching for near optimal solutions.

Legierski (2003) presented a Constraint Programming approach to generate feasible solutions for real university department timetabling data which contains 223 courses. A local search algorithm is used to find better solutions. The approach shows promising results.

Sheau et al. (2009) investigated the constraint-based reasoning algorithm in solving real world course timetabling problems. The approach used constraints-based reasoning as an improvement algorithm based on the number of students for each lecture. The proposed algorithm has been tested on the data taken from the Faculty of Computer Science and Information System, University of Technology Malaysia and the Faculty of Science, Ibb University, Yemen.

C. Meta-heuristic Methods

Over the last few years, meta-heuristics have proven to be highly useful for approximately solving timetabling problems in practice. The main advantage of these techniques is that they can handle a wide range of constraints and capable to escape from local optima (Glover and Laguna 1993). Meta-heuristic is a computational method that optimises a problem by repeatedly trying to improve a candidate solution with regard to a given measure of quality (Talbi 2009).

Osman and Laporte (1996) defined meta-heuristic as:

"*A meta-heuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.*"

Later, Glover and Laguna (1997) interfused a meta-heuristic as:

"*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.*"

Voβ et al. (1999) also gave a definition for meta-heuristic i.e.:

"A meta-heuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method."

In 2005, a new definition of meta-heuristic appareled in Meta-heuristic Network Website:

"A meta-heuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a meta-heuristic can be seen as a general algorithmic framework which can be applied to different optimisation problems with relatively few modifications to make them adapted to a specific problem."

A lot of the meta-heuristic methods are employed on university course timetabling problems. The interested readers can refer to Lewis (2008) for more detailed description of the different meta-heuristic approaches (like Great Deluge, Tabu Search, Simulated Annealing, etc) published in the past few years.

The following subsections present the literature review on meta-heuristic algorithms that have been applied on the enrolment-based course timetabling problem.

Great Deluge

Dueck (1993) proposed a new meta-heuristic algorithm called as Great Deluge. Great Deluge is classified as a variant of simulated annealing algorithm. Great deluge depends on two parameters i.e. the estimate of the quality of solution that a user requires and the computational time that the user wishes to "spend" (Burke et al. 2004). Improving solutions are always accepted. A non improving solution is adaptively accepted if its objective function is less than or equal to some given upper boundary value β (called a "*level*" in the paper by Dueck (1993)) for a minimisation case. The "*level*" is iteratively decreased during the improvement process, by a constant β where β is a decay rate.

```
Great Deluge
Begin
      Sol:=initial solution;
      Set initial water level, level;
      Set final water level, L_final;
      Calculate the decay rate β;
      Do while (stopping criterion)
            generate an Sol*∈N(s);
            if f(Sol*) ≤ f(Sol
               Sol=Sol*;
               level = level- β;
        else
           if f(Sol*) ≤ level
             Sol=Sol*;
        end do;
end;
```

Figure 2.2 A great deluge algorithm (Dueck 1993)

Figure 2.2 shows the pseudo-code of the standard Great Deluge. After initialisation the parameters (initial solution, *Sol*, initial water level, *level*, and the decay rate β). The algorithm starts iteratively generate a neighbourhood solution, *Sol**, by modifying the initial solution, *Sol*. If there is an improvement, the neighbourhood solution will be accepted and replaced with *Sol*, *Sol=Sol**. A worse solutions can be accepted if the quality of neighbourhood solution is less than or equal to *level*. *Sol* will

be updated and *level = level-β.* This process is repeated for a pre-determined number of iteration.

Burke et al. (2002) applied the great deluge algorithm to course timetabling by means of 20 instances of the International Timetabling Competition 2002 which was sponsored by PATAT IV conference in 2003. The proposed technique is proved to be effective and was able to obtain 8 best known results of the 23 datasets. These results represented the third among 21 participants.

Burke et al. (2004) applied a great deluge to solve standard benchmark datasets. In this work, authors investigated two parameters that affect the performance of great deluge algorithm such as an execution time and a level of solution quality. The experimental results show that the great deluge algorithm is able to obtain a number of best known results. At the time of the algorithm was proposed, it was able to obtain 7 new results out of 20. Finally, the authors showed that in the great deluge algorithm only one parameter needs to be defined, i.e. the amount of the available search time, whilst the level of solution quality is taken from the initial solution. Moreover, they reported that the better results can be obtained by more execution time spent.

McMullan (2007) introduced an extended version of great deluge algorithm for Enrolment-based course timetabling problems. The approach focuses on the main parameter which is the decay rate as it dictates how fast the boundary is reduced and ultimately the condition for accepting worse moves is narrowed. The approach uses a decay rate proportional to 50% of the entire run to force the algorithm to reach the optimal solution. The algorithm is able to obtain solutions in a relatively short amount of time and managed to get 60% improvement on some cases when compared to the best known results in the literature.

Landa-Silva and Obit (2008) employed a great deluge with non-linear decay rate on the eleven benchmark course instances introduced by Socha et al. (2002). The algorithm is an extension of the original great deluge algorithm proposed by Dueck (1993). The difference is that the decompose rate changes on every iteration based on

the current water level but in the original great deluge it is preset. Experimentation reveals that the algorithm is capable of producing new best known results on 4 out of the 11 tested instances.

Tabu Search.

One of the most popular meta-heuristic algorithms is Tabu Search, which is proposed by Glover (1986). Glover and Laguna (1997) introduced tabu search as:

"*A meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality*"

The basic idea of tabu search is to explore feasible regions of search space by a sequence of moves. Thus, it can escapes from local optima. Tabu list, *Tl* (can be short or long term memory) contains some forbidden moves to prevent cycling of neighbouring solutions. An aspiration criterion can also be used which change forbidden move to non forbidden if it results in a solution that has an objective value better than the best solution found so far.

```
Tabu Search
Begin
      Sol:=initial solution;
       Set Sol as a trail solution;
       Set Solbest as best solution;
       Set tabu list length;
       repeat
           give N(s), tabu list Tl, aspiration criterion, generate an Sol*;
           find the best Sol*;
           insert moves generated Sol* in Tl;
           if f(Sol*) < f(Solbest then Solbest=Sol* and Sol=Sol*;
              update Tl;
       until (stopping criterion);
end;
```

Figure 2.3 A basic tabu search algorithm (Talbi 2009)

Figure 2.3 illustrates a basic tabu search algorithm for a minimisation problem, where *Tl* is subset of *moves* contributed generating *Sol** and *N*(*s*) represents the set of

neighbouring solutions. Tabu search start with initial solution, *Sol*, then initialise the tabu parameters, tabu list length, *Tl*. Next, several neighborhoods solutions, $Sol^* \leftarrow N(s)$, of the current solution are evaluated. The best non-tabu neighbor solutions is accepted and added into tabu list *Tl*. However, it is already in the tabu list, it can be accepted if its objective function is better than the best one found so far (*aspiration criterion*). Then next, update the best solutions (*Solbest*).

Since 1986 up to date, the concept of tabu search has been applied in both artificial intelligence and optimisation fields. In addition, Tabu Search algorithm has been successfully applied to university course timetabling problems. The first implementation for real university course timetabling problem using tabu search algorithm was by Glover and Laguna (1993). The results reported were good.

Tabu search was improved by many researchers to become one of the preferred solution approaches (Glover and Laguna 1997). Later, Costa (1994) applied two tabu lists, where the first list is for courses and the second list is used to keep track the history of courses and timeslots. This mechanism leads to control the movement of generating the new solutions, i.e. if a course is already inserted into the first tabu list; this indicates that it cannot be used in generating a solution. The second tabu list controls the assignment of timeslots to courses, as the timeslot cannot be assigned for a course while this pair of course and timeslot is inside the second tabu list. A weight mechanism is used to enhance the diversification through the search process. Although this approach obtained good results, but several parameters need to be fine tuned such as tabu list size and weights.

A variable tabu list size has been applied to solve a large high school timetabling problem by Schaerf (1996, 1999). The author inserts each applied move to a tabu list. The weakness of this algorithm is that the size of tabu list is randomly selected. However, experimental results show that this approach is able to schedule most of the courses, thus obtaining better results than the results obtained by manual scheduling.

that have been used by the great deluge proposed algorithm for all datasets. The *x*-axis represents the dataset while the *y*-axis represents the frequency of the neighbourhood structures employed throughout the search. It can be clearly seen that in most of *small* and *medium* instances, neighbourhood structures Nbs_2 and Nbs_3 are the most popular structures used, whereas the popular structures for the *large* dataset are Nbs_1 and Nbs_2. This shows that different size or complexity of the problem might need different neighbourhood structures in order to help the search algorithm to explore the search space. It is also evident that some neighbourhood structures do not contribute much to obtain a good quality solution (with respect to the objective function).

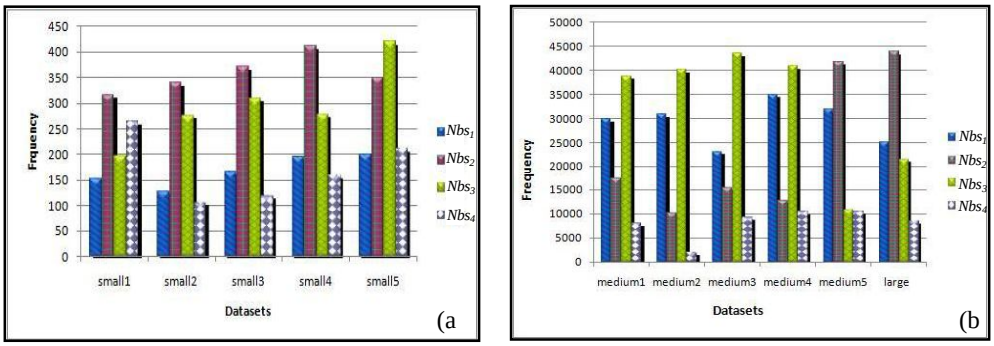

Figure 4.6.(a), (b) Frequency of the neighbourhood structures used for small, medium and large datasets

Table 4.1 shows the results comparison on different neighbourhood structures when applied to the great deluge algorithm for 100,000 iterations. The preliminary results demonstrate that by using 2 neighbourhood structures, the algorithm is still able to achieve better results than 4 neighbourhood structures on all of the cases. Less number of good neighbourhood structures still able to help the algorithm to find better solutions as the algorithm have extra time to explore the search space.

Table 4.1 Comparison on different moves

Data set	Initial solution	The algorithm with 4 Nbs	The algorithm with a set of 2 Nbs
small1	266	0	0
small2	225	0	0

Continue…

…Continue

small3	289	0	0
small4	209	0	0
small5	382	0	0
medium1	957	186	182
medium2	896	225	202
medium3	957	226	223
medium4	835	178	170
medium5	711	167	135
large	1613	980	845

B. Results under More Computational Resources

In this experiment, the potential of the search process is carried out by the algorithm with a relaxed stop condition. For this purpose, the termination criterion is set as 200000 iterations with a different set of moves, as presented in the preliminary experiment. The best results out of 11 runs obtained are presented. Table 4.2 shows the comparison of the approach in this work with other available approaches in the literature on all instances. In Table 4.2 we mentioned "Many methods" and "Many authors", that mean there are many methods proposed by many authors obtained zero penalty cost for the small datasets (i.e. Genetic algorithm and local search by Abdullah and Turabieh (2008), Randomised iterative improvement algorithm by Abdullah et al. (2007a), Graph hyper heuristic by Burke et al. (2007a), Variable neighbourhood search with tabu by Abdullah et al. (2005), Hybrid evolutionary approach by Abdullah et al. (2007b), Non linear great deluge by Landa-Silva and Obit (2008), The fuzzy multiple heuristic algorithm by Asmuni et al. (2005), The max-min ant algorithm by Socha et al. (2002), and The tabu hyper-heuristics by Burke et al. (2003b)).

Note that the best results are presented in bold. It can be seen that the proposed approach in this work has better results on all datasets except for the *large* dataset.

Table 4.2. Best results and comparison with other algorithms under a relaxing stop condition

Dataset	Great Deluge with Tabu search	Best Known results		
	Penalty cost	Penalty cost	Method	Author
small1	**0**	**0**	Many methods	Many authors
small2	**0**	**0**	Many methods	Many authors
small3	**0**	**0**	Many methods	Many authors
small4	**0**	**0**	Many methods	Many authors
small5	**0**	**0**	Many methods	Many authors
medium1	**78**	80	Great Deluge	McMullan (2007)
medium2	**92**	105	Great Deluge	McMullan (2007)
medium3	**135**	**135**	Electromagnetic	Turabieh et al. (2009)
medium4	**75**	79	Electromagnetic	Turabieh et al. (2009)
medium5	**68**	73	Harmony search	Al-Betar et al. (2010)
large	556	**424**	Harmony search	Al-Betar et al. (2010)

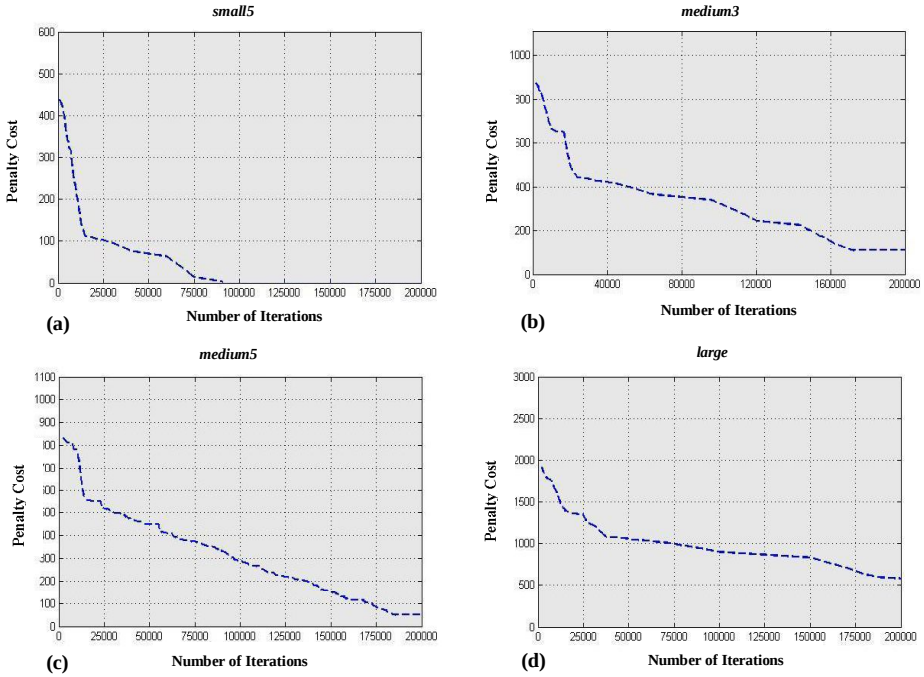

Figure 4.7 Convergences of (a) *small*5, (b) *medium*3, (c) *medium*5 and , (d) *large* datasets using a Great Deluge with Tabu Search algorithms

Figures 4.7 (a), (b), (c) and (d) show the convergence of the search process for

medium3, *medium5* and *large* datasets. The *x*-axis represents the number of iterations, while the *y*-axis represents the penalty cost. It can be seen from the figures that the penalty cost decreases toward the optimal value which is zero. The penalty cost can be quickly reduced at the beginning of the search. This is due to the fact that at the beginning of the search process, the search space is explored more than at the end, which helps to increase the diversity of the search and gives a greater chance to find better solutions.

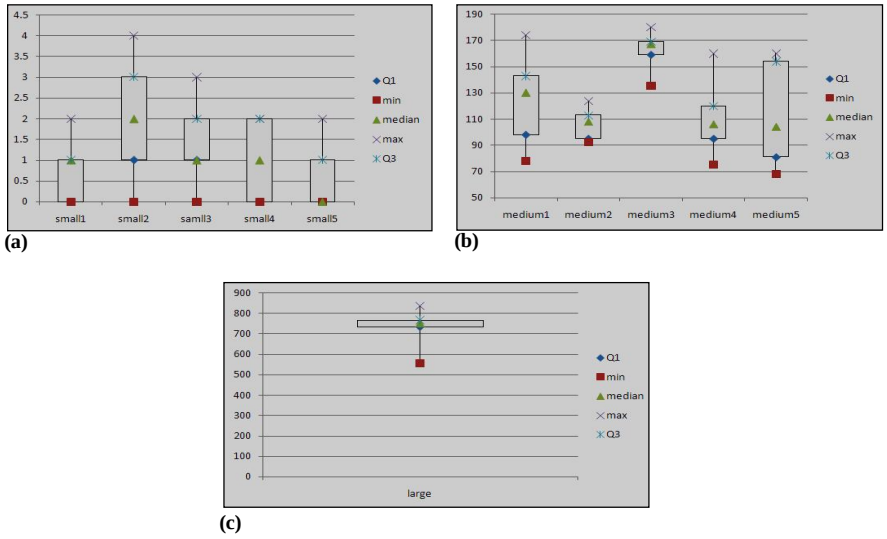

Figure 4.8 Box plots of the penalty costs for (a) *small*, (b) *medium* and (c) *large* datasets.

Figure 4.8 shows the box plots of the penalty cost when solving *small*, *medium* and *large* instances, respectively. The results for the *large* dataset are most consistent (less dispersed) compared to *medium* and *small* datasets (worse dispersed case in these experiments). We believe that the neighbourhood structures (Nbs_1 and Nbs_2) that are applied to the *large* dataset are able to force the search algorithm to diversify its exploration in the solution space by moving from one neighbourhood structure to another even though there may be fewer and more sparsely distributed solution points in the solutions space since too many courses are conflicting with each other. When we compare between *small* and *medium* datasets, Figure 4.8 (b) shows less dispersion of solution points compared to Figure 4.8 (a). Again, applying the same neighbourhood structures (Nbs_2 and Nbs_3) for both instances most likely does not result in similar

behaviour of the search algorithm. This is supported by Figure 4.8 (a) where the dispersion of solution points for *small* datasets is not consistent from one to another. For example, *small2* in Figure 4.8 (a) shows worse dispersion compared to *small4*. From these experiments, we believe that the size of the search space may not be dependent on the problem size due to the fact that the dispersion of solution points are significantly different from one to another, even though the problems are from the same group of datasets with the same parameter values.

It is believed that evaluating results is an important part of research. The standard deviation (StD) is used to evaluate the results in this thesis. It is a widely used measurement of diversity used in statistics theory. It shows how much dispersion there is from the average. A low standard deviation indicates that the results tend to be very close to the average whereas high standard deviation indicates that the results are spread out over a large range of values.

Here, the best, average and worst results out of 11 runs obtained are presented. Table 4.3 shows the best, and average and maximum (Max) of each datasets and also the standard deviation (StD) for all datasets. As shown in Table 4.3, there is a wide range of StD values. For *small* datasets the StD values are between 0 and 1.5. Note that even the value of the StD is small, that normally represents a less dispersion on the solutions. For *medium* and *large* datasets, these imply a big dispersion between the solutions. From this, it can be concluded that the proposed algorithm here works well for the *small* datasets only, but not on larger size of datasets. Thus, motivate to further investigate either the neighbourhood structures or the search algorithm can be modified to tackle this drawback. This will be carried out in the next chapter (Chapter VI)

Table 4.3 Results using Great Deluge and Tabu Search on enrolment-based course timetabling problems

Datasets	Best	Average	Max	StD
small1	0	2	2	0.75
small2	0	1.2	4	1.44
small3	0	1.8	3	1.03
smalll4	0	1	2	0.83

Continue…

... Continue

small5	0	0.6	2	0.68
medium1	78	132.2	174	31.33
medium2	92	114.6	124	10.64
medium3	135	162.0	180	13.46
medium4	75	111.2	160	24.93
medium5	68	113.1	160	34.33
large	556	738.6	835	74.48

4.3.2 Curriculum-Based Course Timetabling Problem

A. UD1 Dataset

The Curriculum-based timetabling problem consists of the weekly scheduling of lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curricula of the university.

Here, only the three first soft constraints outlined in Section 3.3.2 are taken into consideration, as in formulation of curriculum-based course timetabling problems Track 3 UD1 datasets (De Cesco et al. 2008). Table 4.4 shows the results obtained and comparison with the best known solutions. We also compared our results with the best uploaded to the Curriculum-Based Course Timetabling (CBCTT) website[1].

Table 4.4 Best results and comparison with other algorithms.

Dataset	Great Deluge with Tabu search		Abdullah et al. (2009)		Cesco et al. (2008)	Best uploaded to CBCTT	Method applied
	Best	Ave.	Best	Ave.			
comp01	***4***	4.72	***4***	6.8	***4***	**4**	Tabu Search
comp02	22	33.6	*20*	27.6	35	**20**	Tabu Search
comp03	47	31.09	*41*	48.3	52	**38**	Tabu Search
comp04	21	23.7	*20*	21.3	21	**18**	Tabu Search
comp05	261	264.7	*235*	237.8	244	**219**	Tabu Search

Continue…

[1] http://tabu.diegm.uniud.it/ctt/

…Continue

comp06	38	39.3	*24*	24	27	**16**	Mathematical programming
comp07	*10*	11.2	12	13.2	13	**3**	Mathematical Programming
comp08	*22*	22.3	*22*	22.8	24	**20**	Mathematical Programming
comp09	73	74.9	71	75.2	*61*	**54**	Tabu Search
comp10	14	15.3	13	14.8	*10*	**2**	Mathematical Programming
comp11	***0***	0	***0***	5.8	***0***	**0**	Tabu Search
comp12	297	300.6	*261*	265.1	268	**239**	Tabu Search
comp13	51	55.8	67	69.3	*38*	**32**	Tabu Search
comp14	34	36	36	36	*30*	**27**	Tabu Search
comp15	69	69.3	*39*	34.2	46	38	Tabu Search
comp16	50	50.5	30	30	*28*	**16**	Tabu Search
comp17	49	49.7	*35*	37.4	44	**34**	Tabu Search
comp18	82	82.9	*39*	45.2	41	**34**	Tabu Search
comp19	40	45.6	41	47.1	*36*	**32**	Tabu Search
comp20	49	50.2	*19*	19	25	**2**	Tabu Search
comp21	*69*	78.9	88	109.2	*69*	**43**	Tabu Search
DDS1	*87*	88.3			238	**39**	Mathematical Programming
DDS2	***0***	0			***0***	**0**	Tabu Search
DDS3	***0***	2.0			***0***	**0**	Tabu Search
DDS4	***16***	18.1			233	**16**	Tabu Search
DDS5	***0***	0.6			***0***	**0**	Tabu Search
DDS6	***0***	1.6			5	**0**	Mathematical programming
DDS7	***0***	3.36			***0***	**0**	Tabu Search
Test1	***212***	215.8			214	**212**	Tabu Search
Test2	***8***	9.36			***8***	**8**	Tabu Search
Test3	40	46.5			*36*	**35**	Tabu Search
Test4	64	70.9			*43*	**27**	Mathematical programming

The best results out of 11 runs obtained are presented. Note that, the best results (including the results uploaded to CBCTT) are presented in bold. Also, note that the results in italic represent the best result appeared in the literature. It can be seen that, this approach is able to produce solutions with the lowest penalty cost (with respect to the

objective function) on *comp01*, *comp07*, *comp08*, *comp11*, *comp21*, *DDS1- DDS7*, *Test1* and *Test*2 datasets. Generally, in most of the cases, the results presented here are comparable to the performance of Tabu Search (Cesco et al. 2008), Mathematical Programming approach, Genetic Algorithm, and Sequential Local Search algorithms proposed by Abdullah et al. (2009).

Figure 4.9 (a), (b), (c) and (d) show the performance of the algorithm when exploring the search space on *comp01*, *comp08*, *DDS4*, and *Test2* datasets, respectively. Again the *x*-axis represents the number of iterations, while the *y*-axis represents the penalty cost. The distribution of points in these diagrams shows the correlation between the number of iterations and the overall solution quality. An analysis of the diagrams shows that there is a trend of the cost improvement as the number of iterations increases. However, as the number of iterations increases, the slope of the curves illustrates the fast improvement on the quality of solutions at the beginning of the search in all figures where there is possibly many room for improvement. The fast improvement becomes less pronounced towards the end of the search.

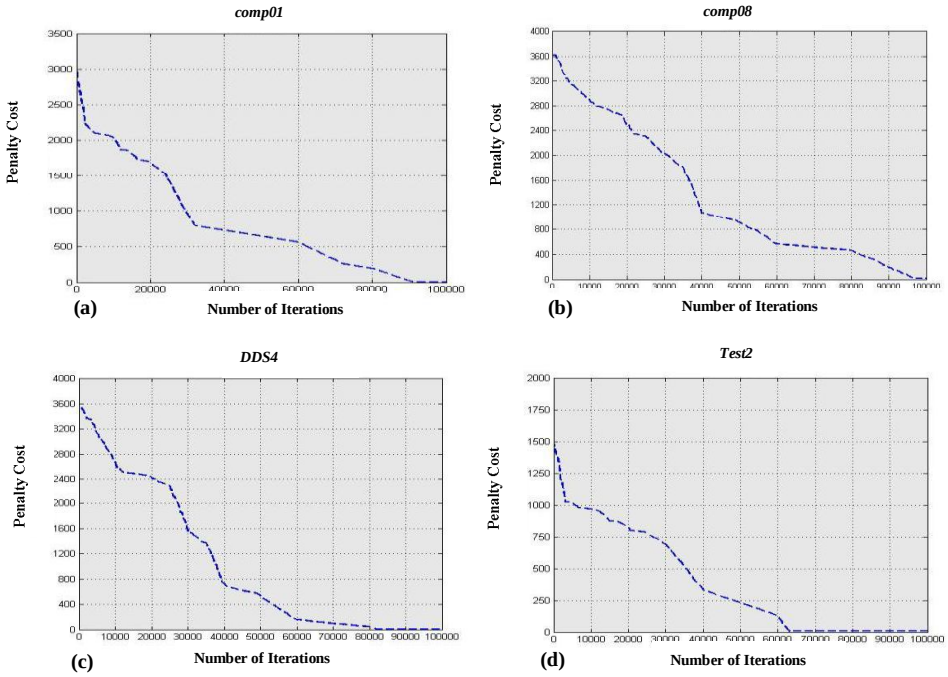

Figure 4.9 Convergence of (a) *comp*01, (b) *comp*08, (c) DDS4, and (d) Test2 datasets using Great Deluge with Tabu Search algorithms

Figure 4.10 shows the box plot of the penalty cost on some of the instances in the UD1 problem considered in this experiment. The results for *comp01*, *comp03*, *comp13*, *comp21*, *DDS4*, *Test2* and *Test3* datasets as in Figure 4.10 show a less dispersion of solution points where the median, the best and the worst are closed to each other. For the *DDS*2 dataset, there is no dispersion at all, which show the robustness of the algorithm especially when tested on *DDS2* dataset. In addition, the median is closed to the best solution compared to the worst (max) on *comp02, comp04, comp06, comp09, comp11, comp14, comp15, comp20, DDS3, DDS5* and *Test1* datasets. This means that 60% of solutions are closed to the best solution. Therefore, we can say that our algorithm is robust and able to produce better solutions on UD1 datasets.

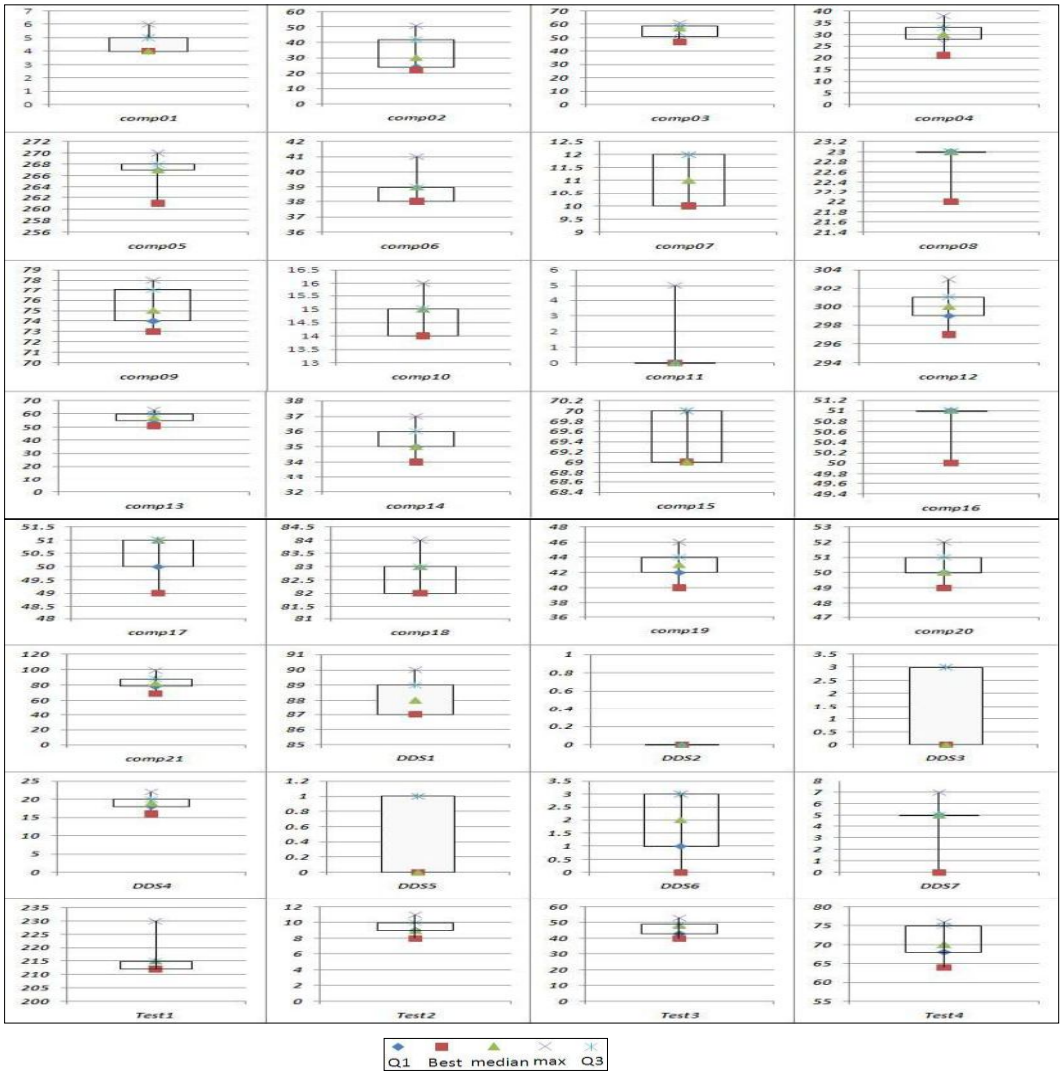

Figure 4.10 Box plots of penalty costs for UD1 datasets.

Table 4.5 summarises the results out of 11 runs obtained under curriculum-based course timetabling problems (UD1 datasets). The best, average, maximum (Max), and the standard deviation (StD) for all datasets are presented. As shown in the Table 4.5, there is less dispersion on the obtained solutions for almost three-quarter of the datasets, where the StD values are between 1 and 2. From this, it can be concluded that the proposed algorithm here is very effective for most of UD1 datasets.

Table 4.5 Results using Great Deluge and Tabu Search on Curriculum -based course timetabling problems (UD1)

Datasets	Best	Average	Max	StD
comp01	4	4.72	6	0.8202
comp02	22	33.6	50	9.4638
comp03	47	31.09	61	5.48386
comp04	21	23.7	48	6.81842
comp05	261	264.7	270	3.29738
comp06	38	39.3	41	0.89442
comp07	10	11.2	12	0.87386
comp08	22	22.3	23	0.52223
comp09	73	74.9	78	1.80403
comp10	14	15.3	16	0.70064
comp11	0	0	0	0
comp12	297	300.6	333	15.6716
comp13	51	55.8	62	3.75136
comp14	34	36.0	37	1.22102
comp15	69	69.3	70	0.50452
comp16	50	50.5	51	0.46709
comp17	49	49.7	51	0.8202
comp18	82	82.9	84	0.75075
comp19	40	45.6	46	1.75809
comp20	49	50.2	52	1.12006
comp21	69	78.9	99	7.64555
DDS1	87	88.3	90	1.12006
DDS2	0	0	0	0
DDS3	0	2.0	3	1.39841
DDS4	16	18.1	22	1.90215
DDS5	0	0.6	1	0.52223
DDS6	0	1.6	3	1.12006
DDS7	0	3.36	7	2.65603

Continue…