HEURISTIC APPROACHES FOR REAL WORLD EXAMINATION TIMETABLING PROBLEMS

By:

Mohd Nizam Mohmad Kahar, MSc

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy
March 2013

Abstract

The examination timetabling (*exam-timeslot-room assignment*) problem involves assigning exams to a specific or limited number of timeslots and rooms, with the aim of satisfying the hard constraints and the soft constraints as much as possible. Most of the techniques reported in the literature have been applied to solve simplified examination benchmark datasets, available within the scientific literature. In this research we bridge the gap between research and practice by investigating a problem taken from the Universiti Malaysia Pahang (UMP), a real world capacitated examination timetabling problem. This dataset has several novel constraints, in addition to those commonly used in the literature. Additionally, the invigilator scheduling problem (*invigilator assignment*) was also investigated as it has not received the same level of research attention as the examination scheduling (although it is just as important to educational institutions).

The formal models are defined, and constructive heuristics was developed for both problems in which the overall problems are solved with a two-phase approach which involves scheduling the exam to timeslot and room, and follows with scheduling the invigilator. During the invigilator assignment, we assume that there is already an examination timetable in place (i.e. previously generated). It reveals that the invigilator scheduling solution dependent on the number of rooms selected from the *exam-timeslot-room* assignment phase (i.e. a lesser number of used rooms would minimises the invigilation duties for staff), this encourages us to further improve the *exam-timeslot-room* timetable solution. An improvement on the result was carried out using modified extended great deluge algorithm (*modified*-GDA) and multi-neighbourhood GDA approach (that use more than one neighbourhood during the search). The *modified*-GDA uses a simple to understand parameter and allows the boundary that acts as the acceptance level, to dynamically change during the search. The propose approaches able to produce good quality solution when compared to the solutions from the proprietary software used by UMP. In addition, our solutions adhere to all hard constraints which the current systems fail to do.

Finally, we extend our research onto investigating the Second International Timetabling Competition (ITC2007) dataset as it also contains numerous constraints much similar to UMP datasets. Our propose approach able to produce competitive solutions when compared to the solutions produced by other reported works in the literature.

Ackowledgements

Thanks God for giving me the strength and patience to finish this challenging journey.

I would like to take this opportunity to express my sincere gratitude to my academic supervisor, Professor Graham Kendall for his guidance, encouragement and constant support throughout this PhD journey.

To my wife (Rosmahyati Dziauddin), my kids, Ikhwan, Nurul, Hazwan and Ridwan who always bring joy and laughter, thanks for being there for me. To my families, abah (Mohmad Kahar Jalani), emak (Rosni Mohd Zain) and my brothers thanks to all of you for giving me indirect support in finishing this work.

Thank you to the Academic Management Office, UMP for all the help in providing the information and the datasets. Last but not least, my thanks go to all my friends.

Thank you...

Publications from this Thesis

1. Kahar M N M and Kendall G (2010). The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution, European Journal of Operational Research, 207 (2): pp 557-565, DOI: 10.1016/j.ejor.2010.04.011.

2. Kahar M N M and Kendall G. Universiti Malaysia Pahang Examination Timetabling Problem: Scheduling Invigilators. Accepted for publication for the Journal of the Operational Research Society, JORS

3. Kahar M N M and Kendall G. A Great Deluge Algorithm for a Real World Examination Timetabling Problem. Accepted for publication for the Journal of the Operational Research Society, JORS

4. Kahar M N M and Kendall G. Solving a real world examination timetabling problem: Multi-neighbourhood great deluge algorithm. Paper ready to be submitted.

Table of Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

1.1 Background and motivation

Every academic institution faces the problem of generating course and examination timetables. Both problems are similar in that we need to assign the courses or exams into available timeslots (Burke, Kingston and deWerra, 2004; Burke et al., 1996) whilst satisfying various constraints. However, the two problems actually differ in terms of the constraints, user preferences and in the way the problem is constructed (Schaerf, 1999; Qu et al., 2009; etc). For example, an exam timetable may allow multiple exams in one rooms unlike a course timetable. This because it is obviously not possible to have two different courses/lectures in the same room. With respect to user preferences, in course timetabling students are free to select their optional courses to suit their own course objectives. This is not the case with an exam timetable as the examinations contain registered students and, therefore we need to consider a clash free (hard constraint) timetable (among others) and student satisfaction (soft constraint) in producing the exam timetable. Course and exam timetables also vary in the way they are constructed, this being the modelling, process environment and scheduling instances (McCollum, 2007). A more detailed discussion on the differences is given in chapter 2.

This work concentrate on the examination timetabling problem. The underlying problem of examination timetabling is considered to be the same (in the basic definition of the problem) as the graph coloring problem and, hence it is an NP-hard problem (Burke, Kingston and deWerra, 2004; Qu et al., 2009, etc). The construction

of an examination timetabling problem is a challenging task and quite often time consuming. It is concerned with assigning exams to a specific number of timeslots so as to satisfy a given set of constraints (Balakrishnan, 1991; Schaerf, 1999, Qu et al., 2009). The constraints that contribute to the complexity of examination timetabling can be divided into two categories, hard constraints and soft constraints. Hard constraints cannot be violated and a timetable is considered feasible if all the hard constraints are satisfied. An example of hard constraint is that no student should be required to sit two examinations simultaneously (i.e. the timetable should be clash free). Soft constraints, on the other hand, are requirements that are not essential but should be satisfied as far as possible, hence it is being used to evaluate the quality of the timetable. An example of a soft constraint could be spreading exams as evenly as possible throughout the exam period. A list of commonly used constraints is given in Qu et al. (2009), Merlot et al. (2003), Burke et al. (1996). In some situations, the problem becomes more difficult as these constraints conflict with one another, where satisfaction of one constraint can lead to a violation of another (Qu et al., 2009). For example, suppose we have a situation where we want to minimise the total examination period and at the same time we wish to spread out exams as much as possible. In such a situation, satisfaction of the first constraint will inevitably lead to poor quality solutions of the second constraint, or vice versa. Moreover, examination timetabling becomes more challenging as the number of student enrolments, courses and constraints increases. In addition room and invigilator constraints add even more complexity to the overall problem in order for the institution to generate a good quality solution whilst satisfying all parties (i.e. administrator, student, lecturer and invigilator). This lead to us a question, is it possible to produce a feasible (and good quality) solution for the UMP capacitated examination timetabling problem considering the individual room capacity and other additional constraints which the UMP system fails to achieve? and is it possible to produce a feasible (and good quality) solution of the invigilator assignment that satisfies the constraints?

A lot of approaches have been investigated in an attempt to produce good quality solutions (as well shall see later in chapter 2). In constructing the examination timetable, three commonly used approaches include creating an examination timetable based on the course timetable, reusing previous exam timetables and creating an

entirely new examination timetable each time one is needed. Much of the work seen in the scientific literature uses the latter approach, with a focus on the search method (see Qu et al., 2009). Such methods include graph colouring, clustering, meta-heuristics, multi-criteria, case-based reasoning, hyper-heuristics etc. A concise description of these methods can be found in Burke and Carter, 1998; Carter and Laporte, 1996; Qu et al., 2009; Petrovic and Burke, 2004. Many research papers on examination timetabling can be found in the PATAT series of conferences (e.g. Burke and Ross, 1996; Burke and Carter, 1998; Burke and Erben, 2001; Burke and De Causmaecker, 2003; Burke and Trick, 2005; Burke and Rudova, 2007).

Examination timetabling problem can be categorised into un-capacitated and capacitated problems. In the un-capacitated examination timetabling problem, room capacities are not considered, while in the capacitated problem the room capacities are considered as a hard constraint, in addition to other hard constraints, e.g. a clash-free timetable (Pillay and Banzhaf, 2008; Abdullah, 2006). According to Burke, Newall and Weare, (1996), the main difficulty in examination timetabling is to obtain a conflict-free schedule within a limited number of time periods and under room availability constraints. Burke et al., (1996) found that 73% of universities reported that accommodating exams is a major problem. Therefore a capacitated problem is considered much more difficult than an un-capacitated problem due to its close resemblence to the real world problem. However, most of the research found in the literature mainly considers the un-capacitated problem (Qu et al., 2009). According to Qu et al., (2009) and Carter and Laporte (1996), most research only addresses a subset of the constraints, involving a few common hard constraints, e.g. no exams with common students assigned simultaneously and size of exams need to be below the room capacity. Similarly, typical soft constraints include spreading conflicting exams as evenly as possible, or not in x consecutive timeslots or days. Most of the research has concentrated on the development of the search methodologies to find a good quality solution (McCollum, 2007; Carter and Laporte, 1996). This has created a gap between the research and practice in which the research does not really mimic the real world problem due to the simplicity of the current problems being tackled by the scientific community (e.g. the lack of substantial benchmark data with a sufficient set

of constraints). Chapter 2 provides a detailed discussion on the exam datasets and the constraints.

In this research, we consider a real-world examination timetabling problem which not only has capacity constraints but also has a number of other constraints which have not previously been investigated in the scientific literature. The additional hard constraints include splitting of an examination into different rooms in the same building and no sharing of rooms among different examinations. The additional soft constraints include room distance of an exam in multiple rooms and the minimisation of the number of rooms an exam can be split across. We also investigate invigilator assignment which is often not done as part of an automated system. A thorough description of the dataset is presented in chapter 3.

1.2 Research scope and objectives

This research is concerned with a real world examination timetabling problem from Universiti Malaysia Pahang (UMP). The UMP timetabling process involves assigning exams to timeslots and rooms, and includes scheduling invigilators. The aim of this research is to construct an exam timetable (*exam-timeslot-room* assignment) for the UMP examination timetabling dataset that has several different features from the existing benchmark datasets and to also construct an invigilator schedule, which has rarely been the subject of research within the scientific community. The UMP *exam-timeslot-room* assignment is a capacitated dataset which contains additional hard constraints in addition to proximity and other commonly used soft constraints. The additional hard constraints are, (a) splitting of an exam into different rooms; the rooms must be in the same building (b) no sharing of rooms between examinations. That is, only one examination paper is scheduled to a particular room. The soft constraints include (a) in the case of a split exam, the distance of the assigned rooms should be minimised (b) the number of rooms for a split examination should be minimised. These constraints have not been investigated before in the literature (as far as the author is aware) even in the Second International Timetabling Competition examination track (ITC2007) which contain more comprehensive constraints than

previous benchmark datasets. A comparison of the constraints is discussed in chapters 2 and 3.

For invigilator scheduling, according to the (UMP) timetable officer, it is difficult to produce a satisfactory invigilation timetable and this has motivated us to investigate the problem. Furthermore, it has not received the same level of research attention as the *exam-timeslot-room* assignment even though it is just as important to the educational institution. A detailed list of all the constraints is described in chapter 3. Currently there is no formal mathematical model and this also motivated us to explore the problem.

In addition to the study of the new dataset, we investigate graph heuristics with candidates lists to construct the examination timetable. An improvement methodology involves a modified extended great deluge algorithm (*modified*-GDA) and a *multi-neighbourhood* GDA approach. The *modified*-GDA is designed with the timetable officer in mind as it uses a simple and easy to understand parameter for ease of operation. Moreover, a comparison with the current solution, which is generated by UMP using some proprietary software is carried out in order to evaluate the effectiveness of the methodology we present against the correct way of generating the timetable. Finally, we investigate the examination track of the Second International Timetabling Competition (ITC2007) using our proposed methodology.

In order to accomplish the above, several objectives are outlined as follows:

1) To compile the *exam-timeslot-room* assignment constraints and compile the *invigilation* assignment constraints.
2) To construct the formal mathematical model for UMP exam and invigilator problem.
3) To implement heuristic methods to generate the *exam timetable* and compare the result with the UMP proprietary software result.
4) To implement heuristic methods to generate the *invigilator timetable* and compare the result with the proprietary software.

5) To implement heuristic methods (as in objective 3) to the ITC2007 datasets.

We hope to provide UMP with an improved examination timetabling construction procedure and we would like to propose our UMP datasets as benchmark problem instances so that the scientific community has access.

1.3 Overview of the thesis

This thesis consists of eight chapters. This chapter presents the background motivation, research scope and objectives. The remainder of this thesis is organised in the following way:

Chapter 2 describes the examination timetabling problem and presents various examination datasets and constraints from the scientific literature. It also presents the current published research on the examination timetabling, reporting the available methods in the literature.

Chapter 3 presents the UMP examination timetabling and invigilator scheduling problem. The constraints are listed along with a description of the UMP datasets that are used throughout this thesis.

Chapter 4 presents the formal model of the UMP examination timetabling problem. Graph heuristics with candidates list are implemented. This method is able to produce good quality solutions compared to the solutions produced from the UMP proprietary software, whilst satisfying all hard constraints which the current system fails to do. The work presented in chapter 4 has been published in the European Journal of Operational Research (Kahar and Kendall, 2010a).

Chapter 5 presents the formal model of the UMP invigilator scheduling problem. The proposed constructive heuristic algorithm is able to produce a good quality solutions when compare to the UMP proprietary software, whilst satisfying all hard constraints which the current system fails to do. Additionally, we include others constraints (on

top of the original invigilation constraints) considering the comments made by the invigilators (Awang et al., 2006). The work is currently under its 2nd review for the Journal of Operational Research Society, JORS.

Chapter 6 presents a *modified*-GDA approach to improve the constructive heuristic solutions. The proposed GDA uses a simple to determine parameter that can find a good quality solution and is able to find a better solution than the initial cost even with a higher desired value (due to it ability to adjust the desired value, boundary and decay rate) while using good neighbourhood heuristics. The modified-GDA approach is able to produce good quality solutions compared to the UMP proprietary software, satisfying all the constraints (which the proprietary software fails to do) and also to improve on the constructive result. Additionally, we also investigate different parameters (i.e. different initial solutions, number of iterations and several neighbourhood heuristics) and carry out statistical analysis to compare the results parameters.

Chapter 7 presents a multi-neighbourhood GDA which is an extension of the work presented in chapter 6. The method uses more than one neighbourhood in order to effectively explore the search space and improve the solution. The multi-neighbourhood simplifies the operation of the algorithm for the timetable officer who does not have to determine suitable neighbourhoods. We show (Kahar and Kendall, 2011) that the choice of neighbourhoods play a major role in a search. The multi-neighbourhood approach is able to generate better quality solution when compared to the *modified*-GDA.

Chapter 8 presents the ITC2007 examination dataset. We implemented the graph heuristics, *modified*-GDA and *multi-neighbourhood* GDA to the ITC2007 examination datasets to determine whether the proposed method able to work with similar problem. The same properties as the UMP examination dataset is used in the experiments comparison with other reported result in the literature shows that the above method able to give a competetive result but it takes a considerable amount of time.

Finally, the overall conclusions of the work presented in this thesis and research directions for future work in this area are presented in Chapter 9.

1.4 Research contributions

The contributions are a summary of the work in chapters 3 to 8. The detailed contributions are discussed in the corresponding chapters. The overall research contributions can be classified in terms of contributions to the scientific community and contributions to institution (UMP).

Contributions to the scientific community:

1) Develop a formal model of the UMP *exam-timeslot-room* timetabling problem that contains new constraints which have never been reported before in the scientific literature (see chapters 3 and 4).
2) An investigation of the *invigilator scheduling* which has not received the same level of research attention as the the *exam-timeslot-room* assignment even though it is important to the educational institution. A formal model of the UMP invigilator scheduling problem was developed including additional invigilator constraints taking into account comments made by the invigilators in Awang et al., 2006 (see chapter 3 and 5).
3) We have utilised graph heuristics that call upon candidate lists for the UMP examination timetabling problem and the ITC2007 datasets. The approach is able to produce good quality solutions within reasonable computational times, when compared to the UMP proprietary software (see chapter 4) and we are also able to generate competitive result for the ITC2007 datasets compare to other research reported in the literature.
4) We have applied the modified great deluge algorithm (*modified*-GDA) to improve on the constructive heuristic solutions for the UMP exam problem and ITC2007 datasets. The *modified*-GDA uses a simple and easy to understand parameter which would benefit a novice user (i.e. timetable officer) to operate

the method. The method is able to produce good quality solutions when applied to the UMP examination problem (see chapter 6).

5) Investigation of the parameter settings which include different initial solutions, the number of iterations and neighbourhood heuristics for the *modified*-GDA. A statistical analysis is carried out to determine whether are there significant differences between different parameters settings. The investigation revealed that the choice of parameter plays an important role in the search (see chapter 7).

6) We have applied the modified great deluge algorithm, which uses more than one neighbourhood heuristic (*multi-neighbourhood* GDA), to the UMP exam problem and the ITC2007 datasets. The *multi-neighbourhood* GDA able to generate good quality solutions when applied to the UMP examination problem and relatively good results for the ITC2007 datasets (see chapter 8).

7) The search technique, and insights gained could be applied to similar exam timetabling problems or other related problems.

Contributions to the Institution (UMP):

8) Compiling the *exam-timeslot-room* and *invigilator* timetable requirements (constraints) which have never been properly documented at UMP.

9) Representation of the UMP examination timetabling problem into a mathematical model which is useful for future assesment of the UMP examination timetable solution.

10) Development of UMP examination timetabling system, which includes assigning exams to timeslots and rooms, and scheduling invigilators.

11) Implementation of *modified*-GDA and *multi-neighbourhood* GDA approach that uses a simple to understand parameter for the timetable officer to easily operate the method.

1.5 Summary

This thesis presents a new examination timetabling dataset from UMP which has different characteristics compared to the benchmark datasets (i.e.Toronto, Nottingham, Melbourne) and other real world datasets (i.e. UKM, UiTM and ITC2007). The capacitated UMP examination timetabling dataset are solved using graph heuristics together with candidate lists, *modified*-GDA and multi-neighbourhood GDA that are able to produce good quality solutions compared to the current UMP proprietary software. The proposed methodology was also applied to the ITC2007 examination dataset. Additionally, we also investigated the UMP invigilator scheduling problem and succesfully produced a good quality solution compare to the UMP proprietary software. Furthermore, a new invigilator constraint was also included in addition to the existing constraints, that, in our opinion, closely resembles the institution needs (i.e. officer, staff and invigilator). This work has closed the gap between research and practice making contributions to both the scientific literature and the institution.

Chapter 2

A Review of Examination Timetabling Problems and Methodologies in the Scientific Literature

This chapter provides details of the fundamental aspects of the research area tackled in this thesis. It describes the general timetabling problem, the related constraints that need to be considered in the problem and the techniques that have been used to solve the examination timetabling problem. This chapter comprises eight sections. Section 2.1 describes the definition of timetabling and a brief discussion of the general timetabling problem. Section 2.2 discusses the classification of university timetabling problems. Section 2.3 provides further details of the examination timetabling problem. The variations of the examination timetabling constraints and objectives experimented within the scientific research are discuss in section 2.4. Section 2.5 describes the difference between the un-capacitated and the capacitated examination timetabling problem. Section 2.6 and 2.7 discuss the *exam-room* assignment problem and invigilator scheduling respectively. Lastly in sections 2.8 and 2.9, we summaries the methodologies that have been applied to examination timetabling problems and we present our conclusions.

2.1 Overview of timetabling

A timetable is an organized list that provides information about certain events that are expected to take place. Timetabling can be classified into several categories which include educational timetabling, personnel scheduling, sports timetabling and transportation scheduling (Qu et al., 2009). Each of these timetabling problems differ in their structure, constraints and requirements (Burke, Kingston and deWerra 2004). Research in timetabling continues to attract the attention of researchers due to additional requirements/constraints that are continually introduced and with the end-user insisting on better and better solutions (Burke et al., 1996). Wren (1996) defined timetabling as:

> *"Timetabling is the allocation, subject to constraint, of a 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"*

Another definition given by Burke, Kingston and deWerra (2004):

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

Based on these definitions (among others), timetabling problems involve allocating events into suitable timeslots and resources whilst satisfying *constraints* with the goal of optimising the *objective function* of the problem. Constraints in timetabling can be divided into two categories: hard and soft constraints. Hard constraints cannot be violated. It is not essential to satisfy soft constraints but they should be satisfied as much as possible. For example in examination timetabling, a hard constraint could be that no student is allowed to take two or more exams at the same time. While soft constraints could include spreading exams as evenly as possible throughout the exam period. The objective function is a mathematical model of the problem where it is used to evaluate the solution quality. Hence it is a function of violated soft constraints. A

weighted penalty value is normally associated with each violation of the soft constraint and the objective is to minimise the total penalty value (Ayob et al., 2007)

2.2 Classification of university timetabling problems

University timetabling problems can be divided into examination and course timetabling problems. Carter and Laporte (1996) and Burke, Kingston and deWerra, (2004) agree that examination and course timetabling both have the same characteristics in the general timetabling problem and the core problem can be considered to be the same. 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"*

Carter and Laporte (1996) defined examinations timetabling as:

> *"The assigning of examinations to a limited number of available time periods in such a way that there are no conflicts or clashes"*

Both course and examination timetabling problems are concerned with avoid assigning students sitting two (or more) courses or exams in the same time period. However, significant differences do exist. These include differences in constraints that must be respected (as mention in chapter 1). Table 2.1 and table 2.2 shows an example of hard and soft constraints for course (Abdullah, 2006) and examination timetabling (Qu et al., 2009) problems respectively. Other examination timetabling constraints can be found in the survey paper of Burke et al. (1996). It is subjective to determine whether a given constraint is a hard or a soft constraint. This is because it is entirely dependent on the requirements of the institution.

Besides the differences in constraints, course and examination differ in the way in which they are constructed, which can be divided into *process environment*, *modelling* and *scheduling instances*. In the *process environment,* normally, the course timetable is produced separately and independently by each school, unlike an exam timetable which is usually produced centrally by the academic office (McCollum, 2007; Burke et al., 1996). In *modelling*, for course timetabling, it is constructed based on the projected number of students that will taking the courses, while in exam timetabling it is generated based on the number of registered students on particular course (McCollum, 2007). In *scheduling instances,* exam and courses use different instances although it is from the same source (i.e. courses). Examination timetables are formed based on the offered courses. While, in course timetable we need to schedule the individual lectures, tutorial and labs from the offered course (McCollum, 2007).

Although differences exist between the examination and course problem, the complexity of examination timetabling problem depends on the amount of freedom of choice on students selecting their course timetable (Laporte and Desroches, 1984). The more freedom a student has increases the difficulty in producing a feasible examination timetable. This research focuses on the examination timetabling problem and a comprehensive discussion will follow in the next sections.

Table 2.1 Example of hard and soft constraints for the course timetabling problems (Abdullah, 2006)

Hard constraints
1. A student and a teacher cannot be in two places at the same time.
2. Only one course is allowed to be assigned to a timeslot in each classroom.
3. The classroom capacity should be equal to or greater than the number of students attending the course at a particular timeslot.
4. The classroom assigned to the course should satisfy the features required by the course
Soft Constraints
5. Students should not have a single course on a day.
6. Students should not have to attend more than two consecutive courses on a day.
7. Students should not be scheduled to attend a course that is assigned to the last timeslot of the day

2.3 Examination timetabling

Examination timetabling is an important problem in any educational institution. The solution generated is of great importance and impact to a number of parties including lecturers, students and administrators. Besides the definition, given by Carter and Laporte (1996), many researchers have given their own definition for examination timetabling. Balakrishnan (1991) gives the definition as

> *"The examination scheduling problem typically involves the assignment of exams to specific periods and classrooms in order to obtain a schedule that uses a minimum number of periods and satisfies a number of different objectives"*

According to Schaerf (1999),

> *"The examination timetabling problem requires the scheduling of a given number of exams (one for each course) within a given amount of time"*

Qu et al., (2009) stated that,

> *"Examination timetabling problem involve assigning a set of exams* $E = e_1, e_2 \ldots e_e$ *into a limited number of available timeslots* $T = t_1, t_2 \ldots t_t$ *in such a way that there are no conflicts or clashes"*

Based on the definition above, the examination timetabling is concerned with assigning exams to a specific or limited number of timeslots and rooms with the aim of satisfying the hard constraints (e.g. conflict free timetable) whilst fulfilling the objective (e.g. spread student exams evenly). An example of these constraints is listed in table 2.2.

Table 2.2 Example of hard and soft constraints for the examination timetabling problems (Qu et al. 2009)

Hard Constraints
1. No exams with common resources (e.g. students) assigned simultaneously.
2. Resources of exams need to be sufficient (i.e. size of exams need to be below the room capacity, enough rooms for all of the exams).
Soft Constraints
3. Spread conflicting exams as even as possible, or not in x consecutive timeslots or days.
4. Groups of exams required to take place at the same time, on the same day or at one location.
5. Exams to be consecutive.
6. Schedule all exams, or largest exams, as early as possible.
7. Ordering (precedence) of exams need to be satisfied.
8. Limited number of students and/or exams in any timeslot.
9. Time requirements (e.g. exams (not) to be in certain timeslots).
10. Conflicting exams on the same day to be located nearby.
11. Exams may be split over similar locations.
12. Only exams of the same length can be combined into the same room.
13. Resource requirements (e.g. room facility).

Examination timetabling is known to be equivalent, and therefore as hard, as the graph colouring problem (Burke, Kingston and deWerra, 2004; Carter, 1986). In the graph colouring problem, given an undirected graph $G = (V,E)$, we need to colour the vertices, V of a graph such that no two adjacent vertices share the same colour if there is an edge, E between them (Schaerf, 1999). This problem is formally known as vertex colouring and is an NP-hard problem. The relationship can be described as, with a undirected graph $G = (V,E)$, V is the examination set with v as the number of examinations and E is the edge set in the graph with e as the total number of edges in the graph. Let's say, v_i is the i^{th} examination (i.e. v_1, v_2 etc; see figure 2.1) and students taking both exam v_i and v_j resulting in an edge e_{ij} (i.e. e_{12}, e_{23} etc ; see figure 2.1) with a weight (total conflicting students which cannot schedule the exam in the same timeslot) between the node v_i and v_j. The graph colouring problem, and its relationship to timetabling, is widely discussed in the scientific literature (see for example, de Werra 1997; Burke, Kingston and deWerra, 2004; Schaerf, 1999 and Di Gaspero and Schaerf, 2001).

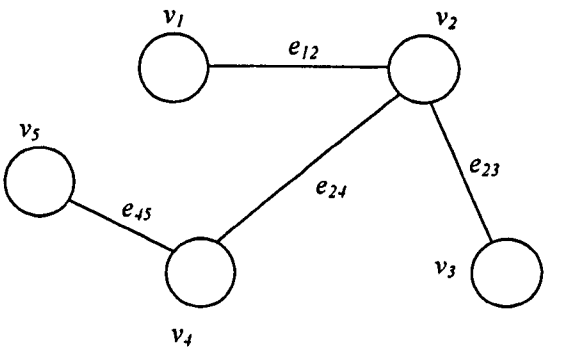

Figure 2.1 Graph colouring

Examination timetabling is considered as *time-consuming*, *difficult* and an *important* task which occurs periodically (i.e. annually, quarterly, etc) in all academic institutions (Carter and Laporte, 1996; Laporte and Desroches, 1984). It is a *time-consuming* process due to the fact that it involves several stages which include data collection, constraint modelling, algorithmic modelling and solution modelling (McCollum, 2007). All of these stages are very important and thus a careful strategy is required. According to Burke et al. (1996), up to 75% of timetables are altered between draft and final versions. The reasons for these alterations include data being made available late, incorrect data and poor quality timetables being generated. A high percentage of the alterations involve late and incorrect data. Therefore a precise and close interaction with all parties (e.g. lecturers and faculty data collection; administrator constraint modelling) should be carried out to avoid any problems. A miscommunication or misinterpretation during the early stages could lead to changes being required in the generated solution. Examination timetable are becoming more *difficult* to generated due to the modular approach which allows students to freely select their courses whilst adjusting their schedule to suit with their own preference. Other factors which further increase the difficulty include the number of examinations being offered, the number of students and constraints (in order to increase student satisfaction) requested by the institution. An example of a new type of constraint (and there are others) involves students from a Muslim background who require Fridays free of examinations (McCollum, 2007; Ayob et al. 2007). Additionally, exams are an *important* part of the overall student coursework assessment and it is normally held at the end of every semester. The solution should satisfy all parties (especially the students) and hence, we need to consider many factors in constructing the timetable whilst ensuring no clashes

for the students, adequate gaps between each exam papers, sufficient marking time for lecturer and administrators being satisfied so that with the timetable less (or no) changes are required (McCollum, 2007).

A lot of approaches have been investigated (Qu et al. 2009) in an attempt to produce good quality solutions (as we shall see later in the following sections). The problem varies from one institution to another (Burke et al., 1996). Every institution has a different set of requirements in order to effectively utilise their resources, meet the requirements of their business, provide a high level of satisfaction to their students etc. Therefore, an examination timetabling system has to be developed to meet these individual requirements.

The examination timetabling problem can be categorised into un-capacitated or capacitated problems. In the un-capacitated, individual room capacities are not considered as the hard constraint, compared to the capacitated problem (Pillay and Banzhaf, 2009; Abdullah, 2006) which does consider individual room capacities. A further discussion on the capacitated and un-capacitated will follow in section 2.6. This research investigates a new capacitated examination timetabling problem using a real world dataset taken from Universiti Malaysia Pahang (UMP). This dataset has never been investigated before in the literature and it has several new constraints in addition to those commonly used in the literature. A detailed discussion on the dataset is given in chapter 3.

2.4 Variations of constraints and objectives investigated in examination timetabling problem

Variations of examination timetabling constraints can be seen in the literature. This because different institutions have different requirements and constraints to suit their business model. Furthermore, the parties affected by the examination timetable would have different preferences for a good quality timetable. For example, an administrator might require that all the exams are to be scheduled and that no student should be assigned to sit two exams at the same time. From a students prespective, they might

prefer that their exams are spread as much as possible to allow for revision time between exam papers. In this section, we consider some of the constraints that are commonly used in the examination timetabling problem. Doing so, we hope to compare the constraints being used by other researchers and the new constraints that arise in this research.

In the examination timetabling research community, the most commonly used datasets are those from Toronto (Carter, Laporte and Lee, 1996), Nottingham (Burke, Newall and Weare, 1996) and Melbourne (Merlot et al. 2003). Among these three dataset, the Toronto dataset has received the most research attention. Many papers, which use this dataset, can be found in the PATAT conference series of selected papers. (i.e. Burke and Ross, 1996; Burke and Carter, 1998; Burke and Erben, 2001; Burke and De Causmaecker, 2003; Burke and Trick, 2005; Burke and Rudova, 2007). Recently the Second International Timetabling Competition (ITC2007) dataset has been introduced by McCollum et al. (2008) which includes more realistic problems than the benchmark problems. Other examination datasets also exist, for example UKM (Ayob et al., 2007) and UiTM (Kendall and Hussin, 2004; Hussin, 2005).

2.4.1 Toronto datasets

The Toronto dataset consists of thirteen real-world exam timetabling problems with three from Canadian highs schools, five from Canadian institutions, one from the London School of Economics, one from King Fahd University, Dhahran and one from Purdue University, Indiana (Carter, Laporte and Lee, 1996). The dataset requires no clashing and to spread student examination. The dataset can be downloaded from *ftp://ftp.mie.utoronto.ca/pub/carter/testprob/*. Table 2.3 show the information of the Toronto datasets. Qu et al. (2009) classified the problem instances into I and II to allow genuine comparison between the scientific community.

Table 2.3 Toronto datasets (Qu et al., 2009)

Problem Instance	Exams	Students	Enrolments	Conflict Density	Timeslots
car91 I	682	16925	56877	0.13	35
car91 II	682	16925	56242/56877	0.13	35
car92 I	543	18419	55522	0.14	32
car92 II	543	18419	55189/55522	0.14	32
ear83 I	190	1125	8109	0.27	24
ear83 II	189	1108	8014	0.27	24
hec92 I	81	2823	10632	0.42	18
hec92 II	80	2823	10625	0.42	18
kfu93	461	5349	25113	0.06	20
lse91	381	2726	10918	0.06	18
pur93 I	2419	30029	120681	0.03	42
pur93 II	2419	30029	120686/120681	0.03	42
rye92	486	11483	45051	0.07	23
sta83 I	139	611	5751	0.14	13
sta83 II	138	549	5689	0.14	13
tre92	261	4360	14901	0.18	23
uta92 I	622	21266	58979	0.13	35
uta92 II	638	21329	59144	0.13	35
ute92	184	2749	11793	0.08	10
yor83 I	181	941	6034	0.29	21
yor83 II	180	919	6012	0.29	21

Carter, Laporte and Lee, (1996) introduced the dataset and investigated two variants of the objectives with the aim to minimise the number of timeslots needed and to spread conflicting exam within the timeslots (using proximity values of 16, 8, 4, 2 and 1). They tested all of the datasets using the graph colouring heuristic with clique initialisation and backtracking. Gaspero and Schaerf (2001), investigated the dataset in which they consider the first and second order conflict. First order conflict (hard constraint) is when a student has to take two exams scheduled in the same timeslot, while second-order conflict (soft constraints) is when a student has to take two exams in consecutive periods. They carried out the investigation using tabu search. Several researchers have included other objectives into the original dataset. Burke, Newall and Weare, (1996) consider maximum room capacity per timeslot and second-order conflict of same day constraints. Burke, Newall and Weare, (1998) further modify the

dataset by including a second-order conflict of overnight and tested the dataset on three timeslots a day (Monday to Friday). They also include a total seating capacity constraint in the experiment in addition to other constraints introduced by Carter, Laporte and Lee, (1996).

Merlot et al. (2003) investigated the dataset by using several methodologies that include constraint programming, simulated annealing (SA) and hill climbing (HC). The aim is to minimise the number of timeslots needed, spreading conflicting exams within limited number of timeslots, to minimise second-order conflict of the same day and overnight. Asmuni et al., (2005) investigate the dataset using graph colouring heuristics with fuzzy reasoning to sort the exams. They used the original constraints as in Carter, Laporte and Lee, (1996). Kendall and Hussin (2005) applied tabu search hyper-heuristics that work with high level heuristics (i.e. the search methodology does not deal directly with the solution).

2.4.2 University of Nottingham

The Nottingham dataset were introduced by Burke, Newall and Weare, (1996). It consists of three timeslots a day (Monday to Friday) with a total of 23 timeslots. The dataset uses no clashing and total capacity constraint with the objective to minimise the number of second order conflicts on the same day. Table 2.4 show the information of the University of Nottingham examination dataset. The dataset can be downloaded from *http://www.asap.cs.nott.ac.uk/resources/data.shtml*. In 1999, Burke and Newall investigated a decomposition approach by using graph heuristics (i.e. CD, LD and SD) with the aim to minimise second order conflicts on the same day and overnight. Merlot et al., (2003) also applied the same method as describe previously to the Nottingham dataset. Burke et al. (2004), investigated the dataset using a great deluge algorithm (GDA) using the same objectives that is to minimise second-order conflicts on the same day, as well as overnight.

Table 2.4 University of Nottingham dataset (Burke, Newall and Weare, 1996)

Exams	Students	Enrolments	Conflict Density	Timeslots	Capacity
800	7896	34265	0.03 (3%)	23	1550

2.4.3 University of Melbourne

The dataset from the University of Melbourne was introduced by Merlot et al., (2003). They introduced two different datasets which has two timeslots on each weekday, and the capacity for each timeslot varies. The datasets also includes period exclusive constraints where exams are pre-assigned to specific sessions or can only be held in a limited set of sessions. The aim of the dataset is to minimise second-order conflict on the same day or overnight. These datasets can be downloaded from *http://www.or.ms.unimelb.edu.au/timetabling*. Table 2.5 show the information of the University of melbourne examination datasets. In addition to Merlot et al., (2003), Cote, Wong and Saboun, (2005) investigated the dataset using a bi-objective evolutionary algorithm where tabu search (TS) and variable neighbourhood descent (VND) were utilised.

Table 2.5 University of Melbourne datasets

Problem Instance	Exams	Students	Enrolments	Timeslots
I	521	20656	62248	23
II	526	19816	60637	31

2.4.4 Second International Timetabling Competition (ITC2007) datasets

The second international timetabling competition (ITC2007) is divided into course and examination timetabling. In this work we will focus only on the examination dataset. ITC2007 aims to create a platform for researchers to asses their algorithms on real world timetabling problems. The ITC2007 examination dataset contains the following constraints; no student sits more than one exam at the same time and the exams should not exceed the room capacity. An exam assigned to a timeslot should not violate the

timeslot lengths and the exams need to be comply with a specified arrangement (for example, assign examA after examB, examA must use room15 etc). The objective is to minimise second-order conflicts on the same day, minimise the number of students sitting two exams in a day, minimise mixed duration of exams within a timeslot, minimise the usage of a particular timeslot or room and schedule larger examinations as early as possible. The details of the examination competition track can be found in McCollum et al., (2008). Researchers which have investigated this dataset include McCollum et al., (2009) which uses iterated forward search, hill climbing and great deluge algorithm. Gogos, AleFragis and Housos, (2008) uses a multistage approach that uses GRASP, simulated annealing and mathematical programming. McCollum et al., (2009) applied a two-phase approach with adaptive heuristic ordering as the constructive phase and improved the solution using an extended great deluge algorithm. Table 2.6 show the information of the ITC2007 datasets (examination track).

Table 2.6 Second International Timetabling Competition (ITC2007) datasets

Instance	Conflict Density (%)	Exams	Students	Periods	Rooms	Period HC	Room HC
Exam-1	5.05	607	7891	54	7	12	0
Exam-2	1.17	870	12743	40	49	12	2
Exam-3	2.62	934	16439	36	48	170	15
Exam-4	15	273	5045	21	1	40	0
Exam-5	0.87	1018	9253	42	3	27	0
Exam-6	6.16	242	7909	16	8	23	0
Exam-7	1.93	1096	14676	80	15	28	0
Exam-8	4.55	598	7718	80	8	20	1
Exam-9	7.84	169	655	25	3	10	0
Exam-10	4.97	214	1577	32	48	58	0
Exam-11	2.62	934	16439	26	40	170	15
Exam-12	18.45	78	1653	12	50	9	7

2.4.5 Universiti Kebangsaan Malaysia (UKM) dataset

Beside the standard benchmark dataset, they are several other exam dataset discussed in the literature. Ayob et al., (2007) introduced a capacitated dataset from UKM, Malaysia. The dataset requires all exams to be scheduled. They forbid students taking more than one exams at the same time and sitting three consecutive exams in a day. Exams with a specified room (room exclusive constraint) must be fulfilled and those students assigned to sit consecutive exams must be assigned to the same room. The objectives involve evenly spreading the exams and minimise students having consecutive exams on the same day. Table 2.7 show the UKM dataset and table 2.8 show the room capacity of the dataset.

Table 2.7 Universiti Kebangsaan Malaysia datasets (UKM06-1)

Exams	Students	Enrolments	Timeslots	Capacity
818	14047	75857	42	1550

Table 2.8 Available rooms for dataset UKM06-1

Room	Room Capacity
DPBestari	850
DGemilang	610
Dewan (DECTAR)	610
LobiUtama (DECTAR)	270
PSeni (DECTAR)	152
LobiA (DECTAR)	70
LobiB (DECTAR)	70

2.4.6 Universiti Teknologi MARA (UiTM) dataset

Kendall and Hussin (2004) introduced a capacitated dataset from UiTM Malaysia. The constraints involve scheduling all exams, first order conflict and coincidence constraints (i.e. exams that required scheduling together must be assigned in the same timeslot). The objective is to spread exams as evenly as possible, which is calculated using the proximity value as in Carter, Laporte and Lee, (1996) and penalising exams

that are scheduled during the weekend. Table 2.9 show the information of the UiTM dataset.

Table 2.9 Universiti Teknologi Malaysia (UiTM) dataset

Exams	Students	Enrolments	Timeslots
2,063	84,675	357,761	40

A summary of the constraints and objectives of the datasets describe above are shown in table 3.1, in chapter 3.

2.5 Uncapacitated and capacitated examination timetabling problem

Most of the research in the literature has investigated the un-capacitated examination timetabling problem, concentrating on the algorithm and algorithmic performance in terms of producing solutions effectively and quickly (see Qu et al., 2009). Although un-capacitated benchmark datasets are popular, McCollum (2007) and Carter and Laporte (1996) believe that, researchers are not dealing with all aspects of the problem. That is, they are only working on a simplified version of the examination problems. Qu et al. (2009), in their survey paper, reveal that most research only addresses a few common hard constraints. For example, no exams with common students assigned simultaneously, the size of exams need to be below room capacity etc. Commonly used soft constraints include spreading conflicting exams as evenly as possible, or not in x consecutive timeslots or days.

The capacitated problems on the other hand more closely resemble the real world problem as it includes a room capacity constraint. However, the capacitated problem has received less attention from the research community. This is probably due to the lack of benchmark datasets. Capacitated problems require more comprehensive data as they have to include the room capacity as well as the other data also required for the less complex problem (e.g. student and exam list). This extra information can be difficult to collect (McCollum, 2007). In addition, the capacitated problem is much harder to solve; see Burke et al. (1996) survey paper where 73% of the universities

agree that accommodating exams is a difficult problem. Burke et al., (1996) mention that the difficulties of accommodating exams are because of, firstly, the lack of halls available probably due to its unsuitability for exams or the room is still being used for lecturing. Secondly, is the problem of splitting exams between more than one room which could lead to others constraints (i.e. splitting an exam onto different sites or taking into account between rooms).

Some of the current benchmark datasets lack the relevant information on the seating capacity of each rooms. However, due to the interest of the capacitated problem and making the benchmark dataset more like the real world problem, Burke, Newall and Weare, (1996) made a modification to the benchmark dataset (e.g. Toronto dataset) by including an overall capacity as if all exams were taking place in one big room (e.g. a sports hall). The same goes to Nottingham and Melbourne dataset which is only concerned with the total seating capacity. That is, the total number of students sitting in all exams in the same timeslot must be less than some specified number. However, according to Merlot et al. (2003), this represents a simplified of the problem whereas normally in solving a real-world problem, we would have to take into account individual room capacities, but this obviously depends on institutional requirements. ITC2007 does include individual room capacities (just like the UMP dataset studied here). One difference to the UMP dataset is that, UMP does not allow exams to share a room. However, UMP does allow exams to split across several rooms (unlike ITC2007 that disallows splitting) but restricts the exams being split to be held within the same building and trying to place those rooms as close to one another as possible, this complicates the problem. A further description of UMP constraints is described in chapter 3.

2.6 *Exam-room* assignment problem

The solution approaches seen in literature for the exam timetabling problem can be separated into *exam-timeslot* assignment and *exam-room* assignment. The most popular approach is the *exam-timeslot* assignment. Only a few works have discussed *exam-room* assignment (Carter and Laporte, 1996; Laporte and Desroches, 1984;

Leong and Yeong, 1990). In a survey by Carter and Laporte (1996), the idea of a subroutine is frequently used for the *exam-room* assignment problem in which the rooms and the exams are arranged in decreasing order based on their capacities and on their size respectively. The biggest exam is assigned to the room with the smallest capacity that can fit this exam. If there is no room sufficient to hold this largest exam, then the largest room is fully assigned and the remaining exams are assigned to other rooms. Laporte and Desroches (1984) use a room allocation subroutine to solve the exam timetabling problem. The largest exams are scheduled in rooms with the largest capacity. If the size of the exam exceeds the capacity of the room, the residual is considered as a new size of the exam and the procedure is repeated until all exams are assigned. There is no limit on the number of exams that can be held in the same room. Leong and Yeong (1990) consider the problem of assigning exams to room that minimized the residuals. They limit the number of exams that can be held in a single room to a specified number. Firstly, they try to assign each exam to a single room. If this is not possible the exam is allocated to a neighbourhood cluster.

Based on the discussion above, the un-capacitated and capacitated (with total seating capacity) problem can be solved using a two-phase approach (1. schedule exams to timeslots and, 2. schedule exams to rooms), as both allow more than one exam in a room (sharing room with several different exams). This will provide a feasible solution in the *exam-room* assignment phase as long as the capacity of the room is greater than the number of students (Dammak, Elloumi and Kamoun, 2006). However, if individual room capacities are used, including prohibiting having more than one exam in a room, it does not guarantee that we are able to produce feasible solution using the two-phase approach. A solution repair mechanism might be introduced in order to arrive at a feasible solution.

2.7 Invigilator scheduling

The exam-timetabling problem can be defined as (Reis and Oliveira, 1999):

> *"ET–examination timetabling: Scheduling (in time) of the exams of a set of university courses avoiding overlapping exams having common students and spreading the exams for the students as much as possible. Room assignment and invigilator assignment can be done prior to or after the exam timetabling phase."*

Based on this definition, the whole examination timetabling problem process involves exams, timeslots, rooms and invigilators. However, most of the research found in the scientific literature investigates the *exam-timeslot/room* assignment problem, that concentrate on the algorithmic performance with the aim of producing good quality solutions in minimal time (see Qu et al., 2009). The scheduling of invigilators is often ignored. The Toronto, Nottingham and Melbourne datasets only cover one third of the examination timetabling problem as their focus is on assigning exams to timeslots (although the Nottingham and Melbourne datasets do consider maximum seating capacity in a timeslot). The second International Timetabling Competition (ITC2007) dataset (McCollum *et al.*, 2010) includes more realistic problems than the benchmark datasets but it is still lacking with respect to invigilator scheduling that forms part of the complete educational examination timetabling problem (Burke et al., 1996; Hussin, 2005).

Invigilator scheduling contains many hard and soft constraints which vary greatly from one institution to another. An example of a hard constraint is that invigilators are not assigned to multiple invigilation duties at the same time. A typical soft constraint specifies that invigilation duties need to be evenly spread among the invigilators. Furthermore, in a survey by Burke et al. (1996), it was found that 29% of universities agree that the task of invigilator scheduling is a major problem. This is also reported by Cowling, Kendall and Hussin, (2002) and Ong, Liew and Sim, (2009) where many invigilators are not satisfied with their individual schedule. Additionally, in a survey

on the UMP invigilator schedule, Awang et al., (2006) reported that the invigilators are not satisfied with their schedule and would like a better invigilation schedule.

Currently, there is no recognised dataset for the invigilator scheduling problem in the scientific literature (as far as the authors are aware). In our opinion, invigilator scheduling has been largely overlooked by the scientific community, despite being as important as *the exam-timeslot-room* assignment problem to the institution. Therefore the invigilator scheduling problem seems to be worthy of investigation.

2.7.1 Implementation by academic institutions

In our view, invigilator scheduling can be divided into three categories with respect to the staff that are employed to carry out the invigilations.

I1) Outside staff: the institution hires non-staff (typically these are from outside the institution) to invigilate the exam timetable. This approach reduces the complexity of the problem as we only need to consider fulfilling the requested number of invigilators for each exam/room.

I2) In-house staff: the institution use their own staff to invigilate the exams (Ong et al., 2009). Some insitutions use only academic staff (e.g. lecturers) while others might also include non-academic staff (e.g. administrators, technicians, postdoctoral researchers etc). The academic staff are often assigned as chief invigilators while non-academic staff are assigned to help in the invigilation process. Compared to (I1), this approach may have a significant number of constraints such as invigilators not being able to invigilate their own exam paper (or alternatively being expected to), not being assigned to more than one invigilation duty at a time, the invigilation duties being evenly spread among the staff etc.

I3) Mixed: the institution use their own staff and hire outside staff to invigilate the exam timetable. The mixing of staff types provides flexibility to the institution

as it enables a variety of working methods to be adopted (e.g. in-house staff act as chief invigilators and outside staff to provide the relevant support).

The implementation of I1 would increase operational costs as the institution needs to pay for the outside staff. In our opinion, a mix of outside and in-house staff (I3) gives more advantages and flexibility to the timetabling office compared to using I1 and I2. However, it also comes at increased operational cost. It also reduces the complexity of the problem compared to I2. However, we recognise that every institution operates in different ways and the staffing model that is adopted is dependent on many factors and what is suitable for one institution may not be suitable for another.

UMP only uses its own staff as invigilators (I2). This result in numerous constraints such as the chief invigilators must be a member of academic staffs, staffs are required to carry out a number of invigilations within the exam period etc. A detailed description of the UMP invigilator constraints is presented in chapter 3.

In this work, we solve the UMP examination timetable in two phases: firstly, we schedule the exams into timeslots and rooms simultaneously (Kahar and Kendall, 2010a). We then use the solution from the first phase as an input to the invigilator scheduling phase. The scheduling of exams into timeslots, rooms and lastly the invigilators has been reported as the best sequence in order to produce a good quality solution (Reis and Oliveira, 1999). Our proposed approach to this second phase is presented in chapter 5, but first we describe the problem informally, and present a formal definition in chapter 3.

2.8 Methodologies applied to the examination timetabling problem

The examination timetabling problem has been the subject of active research for more than 20 years, possibly longer. A variety of algorithms have been proposed and tested, which include graph heuristic, meta-heuristic, constraint based methods, hybridisations; as well as many other approaches, in order to produce a timetable. A comprehensive review and survey of the examination timetabling approaches can be

found in Carter and Laporte (1996), Schaerf (1999), and Qu et al., (2009). Many methodologies can also be found in the PATAT conference series of selected paper (i.e. Burke and Ross, 1996; Burke and Carter, 1998; Burke and Erben, 2001; Burke and De Causmaecker, 2003; Burke and Trick, 2005; Burke and Rudova, 2007). Carter and Laporte (1996) divided the techniques used into four categories: cluster methods, sequential methods, constraint-based methods and meta-heuristics. Petrovic and Burke, (2004) added the following categories: multi-criteria, case-based reasoning and hyper-heuristics approache. A more general classification of the methodologies can be divided into trajectory based and population based approaches. The trajectory based methods operate on individual solutions and randomly explores the search space to find a better solution until a stopping criterion is met (Gaspero and Schaerf, 2001). Examples of trajectory based methods include Hill-Climbing (Merlot et al. (2003), Burke and Bykov (2008), Muller (2007) and Kendall and Hussin (2005b)), Tabu search (Di Gaspero and Schaerf (2001), White and Xie (2004), Abdullah, Turabieh and McCollum (2009) and Kendall and Hussin (2004)), Simulated Annealing (Thompson and Dowsland (1996 and 1998), Wright (2001), Burke et al. (2003) and Frausto and Alonso (2008)), Great Deluge Algorithm (Burke and Newall (2003), Burke et al. (2004), Abdullah et al. (2009) and Turabieh and Abdullah (2011)) and Variable Neighbourhood Search (Abdullah, Burke and McCollum (2005) and Burke et al. (2010a)). These algorithms differ from each other in the method that is used to find a neighbourhood solution in the search space. Population based methods operate on multiple solutions and refine each solution to obtain an optimal solution. Examples include Genetic Algorithms (Corne, Fang and Mellish, 1993; Chu and Fang, 1999; Erben, 2001 etc), Memetic Algorithms (Burke, Newall and Weare, 1996; Burke and Newall, 1999 etc), and Ant Colony Optimisation (Eley 2006 and Eley 2007).

2.8.1 Graph heuristics (GH)

The graph colouring problem involve assigning colours to vertices, so that no adjacent vertices have the same colour (also normally referred to as vertex colouring). Graph colouring techniques have been widely used in solving related problems including the examination timetabling problem (Carter, Laporte and Lee, 1996; Burke, Kingston and

de Werra, 2004). In examination timetabling, the exams are represented by vertices and the edges between the vertices represent the hard constraints. The difference is in the soft constraint where it need to be considered separately and evaluated to measure the solution quality. An illustration of timetabling problem as a graph colouring model can be found in de Werra (1985) and Burke, Kingston and deWerra, (2004).

Originally graph heuristics were used on their own to schedule examinations (Carter, 1986). However, in more recent work they have been used to constructs initial solution (i.e. a so called constructive phase), being hybridised with other methods which acted as an improvement mechanism. Graph heuristics are able to generate reasonably good quality solutions in a short computational time and are easy to implement. Graph heuristic involve ordering the exams in some way (typically be how difficult they are to be scheduled). Thereafter, the exams are scheduled one by one into the timeslots. Common ordering strategies are described below:

a) Largest degree (LD): this heuristic takes the exams that have the most conflicts with other exams and schedules them first.
b) Largest weighted degree (LWD): this heuristic is similar to largest degree except that it takes exams that have the most number of students who are involved in the conflict and schedules them first.
c) Largest enrolment (LE): this heuristic takes exams with the largest number of registered students and schedules them first.
d) Saturation degree (SD): this heuristic chooses exams which have the least number of available periods in the timetable that can be selected and schedules them first.
e) Random ordering (RO): randomly orders the exams.

Largest degree (LD) and saturation degree (SD) normally provides better results compared to other ordering strategies (Qu et al., 2009). Other ordering strategies, and their modified variants, is discussed in Carter (1986). Carter, Laporte and Lee (1996), experiment with different ordering strategies on real and randomly generated exam timetabling problems. They consider conflict free and spreading (proximity cost) of the examination timetable. The results indicated that none of the heuristics show large

differences in performance over all of the problems tested. However, the combined use of backtracking and saturation degree yields better schedules in less computational time. Burke, Newall and Weare (1998) investigated the effect of random elements in saturation degree, color degree and largest degree using (a) tournament selection that randomly selects one from a subset of the first exams in the ordered list; and (b) bias selection that selects the first exam from an ordered list of a subset of all the exams. These simple techniques are able to give relatively good results on three of the Toronto datasets.

For the past few years, graph heuristics have evolve and it is being used in different ways such as dynamic ordering strategies (i.e. adaptive ordering different from SD), multiple graph heuristics ordering strategies and even hybridisation of graph heuristics (with other search methods). Burke and Newall (2004) investigated an adaptive ordering strategy that prioritises the exam to be scheduled (during the constructive approach). It uses a weighted ordered list of the examinations to be scheduled, based on individual soft penalties and difficulty to schedule penalties. The ordering of the exams are updated according to the experience obtained with respect to the difficulty of assigning them in the previous iterations. They investigate the approach on the Toronto and Nottingham datasets. The advantage is that it is not dependent on the initial ordering of the exams. Based on the work above, Rahman et al. (2009) include the concept of squeaky wheel optimization (that is an iterative greedy approach) that consists of constructer, analyzer and prioritizer. Each examination has a priority determined by the chosen graph heuristic, which is dynamically updated during the construction phase. An exam is given more priority in the next iteration if the exam is considered difficult to schedule in the current iteration. Experiments were carried out on the Toronto dataset and the approach is able to produce comparable solutions to other approaches.

Asmuni et al. (2005 and 2009) investigated ordering of the exams based on graph heuristics with fuzzy logic to evaluate the difficulty when ordering the exams on the Toronto datasets. Asmuni et al. investigate the combination of two (Asmuni et al., 2005) and three (Asmuni et al., 2009) graph heuristics to guide the order in which exams are selected to be scheduled. They also investigate the effect of computational

time, number of *skipped exams*, and the number of times a *rescheduling procedure* is required. Experiments were carried out on the Toronto dataset and they produce the lowest penalty than any previously published constructive method. The fuzzy combination of SD and LE obtained a good overall performance in terms of low penalty cost. However, different fuzzy functions need to be used on different problems in order to obtain the best results.

Qu and Burke (2009) investigated the used of graph colouring heuristics within hyper-heuristic (HH) methodology in which the HH is used to choose the graph heuristic for constructing the timetables. This is motivated by the fact that graph heuristics, on their own, are not always appropriate methodologies for addressing complex timetabling problems and for some of the problem instances they failed to even generate feasible solutions. However, recent research has shown that they are effective as producing initial solution for meta-heuristics (e.g. Muller (2008), Abdullah, Burke and McCollum (2005) etc.) A further discussion on HH will follow in the following section.

We have implemented graph heuristics for the UMP examination timetabling problem (our case study dataset). The solution produced is superior compared to the software currently used in UMP. In addition, the proposed algorithm adheres to all the hard constraints which the current software fails to do. Chapter 4 gives a detailed description of our methodology.

2.8.2 Hill Climbing (HC)

Hill climbing (or simple descent) is a classic local search technique. In each iteration, the candidates solution, s' is selected at random from the neighbouring solution, $N(s)$. The candidate solution s' is accepted, and replaces the current solution s, if $f(s')$ is an improvement compared to $f(s)$ (see figure 2.2). Hill climbing is simple and easy to implement. However the disadvantage is that, it is easily trapped in local optima. Therefore, researchers tend to hybridise hill climbing with other search methods such as meta-heuristic methodologies (e.g. evolutionary algorithms, simulated annealing etc). For example, Merlot et al. (2003) incorporated a multi-stage search method to

solve an examination timetabling problem that included constraint programming, simulated annealing and hill climbing. Burke, Newall and Weare (1996) hybridised a genetic algorithm with hill climbing to further exploit the individual solution. This hybridisation method often referred to as a memetic algorithm which will be discuss further in the following section. Kendall and Hussin (2005b) applied a hyper-heuristic and hill climbing to the examination timetabling problem. Muller (2007) uses hill climbing in combination with a great deluge algorithm (as well as simulated annealing) for the ITC2007 problem. Recently, Burke and Bykov (2008) propose a late acceptance strategy for the hill climbing. The method delays the comparison step between candidate solution and current (best) solution. The late acceptance hill climbing is able to produce a good quality solution compared to other works for the Toronto datasets. An improved method have also been formulated based on hill climbing, in order to try and counteract its disadvantages (i.e. escaping from local optima). Tabu search is just one example of such a method. This will be discussed in the next section.

```
sinceLastMove := 0
While sinceLastMove < 1,000,000 do
Choose exam e and period t at random s.t. t != period(e)
    If penalty(e, t) ≤ penalty (e, period(e)) then
    Move exam e to period t
        sinceLastMove := 0
    Else
        sinceLastMove+ = 1
    Endif
Done
```

Figure 2.2 Hill climbing procedure (Burke and Newall, 2002)

2.8.3 Tabu search (TS)

Tabu search proposed by Glover (1986) works in a similar way to hill climbing but incorporates a memory to encourage exploration of the search space (diversification). Glover and Laguna (1997) define tabu search as:

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

Tabu search starts from an initial solution s_0. The search will iteratively explore a subset $N'(s)$ of the neighbourhood $N(s)$ where s is the current solution. The search encourages exploration by accepting neighbourhood solution with the lowest value, even though (assuming minimisation) its value maybe worse than the current solution. Accepting a non-improving move will allow the search to explore areas beyond local optima. However, having to choose a solution from a subset of solution $N'(s)$ will typically lead to cycling. To prevent the search from becoming stuck in a local optima, a memory (called tabu list) is used to hold recently selected solutions (or, more usually, their attributes) and these moves (stored in tabu list) are forbidden to be performed for a certain number of iterations (depending on the tabu list size). However, a mechanism called the aspiration criterion can be used to make a solution tabu free if the resultant evaluation gives a good quality solution (typically better than the best solution seen so far). Figure 2.3 show the tabu search procedure.

Step 1. Choose an initial solution i in S. Set $i^* = i$ and k = 0.

Step 2. Set $k = k+1$ and generate a subset V^* of solution in $N(i, k)$ such that either one of the tabu conditions $t_r(i, m) \in T_r$ is violated ($r = 1,...,t$) or at least one of the aspiration conditions $ar(i, m) \in Ar(i, m)$ holds ($r = 1,...,a$).

Step 3. Choose a best $j = i \oplus m$ in V^* (with respect to f or to the function $\tilde{f}$) and set $i = j$.

Step 4. If $f(i) < f(i^*)$ then set $i^* = i$.

Step 5. Update tabu and aspiration conditions.

Step 6. If a stopping conditions is met then stop. Else go to Step 2.

Figure 2.3 Tabu Search procedure (Hertz, Taillard and deWerra, 1995)

Di Gaspero and Schaerf (2001) experimented with a shifting penalty and violation mechanism on examination timetabling. The shifting penalty mechanism uses a varying weight on the constraints (hard and soft) to encourage exploration of the solution space. They consider (a) violation of either hard or soft constraints, or (b) violation of hard constraints only. These two features combined with a variable-size tabu list and a good quality initial solution was able to give a good quality solution. In

2002, Di Gaspero applied a combination of tabu search with multiple neighbourhoods. The approach involves optimising the objective function (*recolor*), perturbing the current solution (*shake*) or obtaining more improvement (*kick*). The *recolor* and *shake* algorithm were applied in sequence until no further improvement, and the algorithm continued with a *kick*. The algorithm outperforms a basic tabu search with a single neighbourhood.

White and Xie (2001) implemented a tabu search algorithm on two of the Toronto datasets (Carter, Laporte and Lee, 1996). They use recency-based short-term and frequency-based longer-term tabu list to prevent cycling and to encourage exploration of the search space. Tabu relaxation were also included during the investigation. The results show that the approach with longer-term tabu lists produces competitive results when compared with other algorithms. In 2004, White et al. further applied the approach to the rest of the Toronto datasets. The results show that the longer-term tabu list improves the quality of the solution and they claim that tabu relaxation is a good strategy because it helps to drive the solution into new areas of the search space.

Wilke and Ostler (2008) applied tabu search to the school timetabling problem. They compare several other methods (i.e. simulated annealing, genetic algorithm and branch & bound) in order to provide a software framework that is capable of solving various timetabling problem. Simulated annealing, generally produced the best result, but tabu search was able to produce an improvement solution in minimal time. Mushi (2006) implemented a tabu search algorithm that creates course timetables by heuristically minimising penalties over an infeasible solution. They investigated a dataset from the University of Dar-as-salaam and compared the results with a manually generated timetable. A two move strategy was used, with an aspiration criterion. The algorithm terminates if there is no improvement after 1000 iterations. Their proposed system performs better than the manual system.

Tabu search (or it features) has been hybridised with other methods. For example, Abdullah, Turabieh and McCollum (2009) hybridise tabu search with memetic algorithms. The tabu list is used to hold the neighbourhood structures that are unable to generate better solutions after the crossover and mutation operation. Neighbourhood

structures that give an improvement were continuously used until no further improvement could be obtained. The approach is able to produce good quality solutions on four of the Toronto datasets. Abdullah and Turabieh (2012) then extended the above work by investigating multi-neighbourhood structures. Other hybridisations of tabu search includes Abdullah, Burke and McCollum (2005), which includes a tabu list implementation within the VNS methodology. Kendall and Hussin (2004) investigated tabu search hyper-heuristics from one of the author's institutions. Frausto and Alonso (2008) hybridised simulated annealing and tabu search.

2.8.4 Simulated Annealing (SA)

Simulated annealing (SA) was proposed by Kirkpatrick in 1983. It was motivated from the physical annealing process of heating up a solid to a high temperature and slowly cooling it down until it crystallises and no further changes occur. For each material, the cooling schedule was very important. Simulated annealing starts from an initial solution (generated using a constructive heuristic) and it will always accept an improved solution, while worse solution are only accepted with a certain probability $P = e^{-\alpha/t}$ where α is the difference between the objective value of the incumbent solution and the objective value of a candidate solution. t is a temperature parameter that decreases as the search progresses, according to some cooling schedule. Figure 2.4 show the simulated annealing procedure. According to Thompson and Dowsland (1998), the cooling schedule has a large influence on the quality of the final solution. Faster cooling schedules tend to lead the search to converge to a local optima, while a slower cooling schedule generally produces a better quality solution but increases the search time. A geometric cooling schedule, after a given number of moves (i.e. specified or successful moves), is often used to reduce the temperature during the search. A tutorial on simulated annealing can be found in Burke and Kendall (2005).

```
T := T0
While T > Tmin do
Choose exam e and period t at random s.t t != period(s)
    If penalty (e, t) ≤ penalty (e, period (e)) then
        Move exam e to period t
    Else
        Move exam e with probability
        Exp((penalty(e, period(e)) – (penalty (e, t))/T)
    Endif
    T := T
Done
```

Figure 2.4 Simulated annealing procedure for minimisation (Burke and Newall, 2002)

SA has been successfully applied to many areas, among them being examination timetabling. Thompson and Dowsland (1996 and 1998) solve the exam timetabling problem in two phases, constructive (finding a feasible solution) and improvement (improving the solution quality) phases. An adaptive cooling schedule was used and the results show that it outperformed a simple geometric cooling approach. In 1998, Thompson and Dowsland further experimented with different cooling schedules and neighbourhood moves. The results show that the kempe chain neighbourhood gives the best quality solutions. The reason being is because of its ability to allow a large number of examinations to move, thus making a significant improvement to solution quality.

Wright (2001) presents sub-cost guided search with simulated annealing to solve school timetabling problems. The sub-costs incorporated into simulated annealing were used to modify the standard probability function of accepting worse solutions by using an *adjusted cost increase* in the probability formula. Experimental results show that the additional feature method significantly improves the results of the simulated annealing method. Burke et al. (2003) applied simulated annealing to the examination timetabling problem to study its behaviour. Their aim is to develop a measure of similarity between examination timetabling problems. The motivation of their experiments is that if the meta-heuristics works well on the problems, therefore the problem would be *similar*. Hence, a different problem can be solved (effectively) by determining the similarity between the problem and the appropriate search method.

SA is used to produce solutions from the Toronto datasets (Carter, Laporte and Lee, 1996).

Frausto and Alonso (2008) hybridise simulated annealing and tabu search algorithms to solve the Post Enrolment Course Timetabling (track 2) from ITC2007. The method is divided into two phases; constructing a feasible solution and an improvement phase. In the first phase, SA is use to produce a feasible solution. Additional timeslot were included during this phase. In the second phase, SA is used to find a solution as close to the optimum as possible, within a specified time limit. However, if SA shows no improvement during this stage then, the algorithm will continue with tabu search (only if the time limit permits). The algorithm was able to successfully produce feasible solutions although it was lacking in overall solution quality.

Zhang et al. (2010) applied SA to the high school timetabling problem. They proposed a new neighbourhood structure that swaps exams between pairs of timeslots. The new neighbourhood structure increases the efficiency and performance of simulated annealing. The computational results show that the proposed heuristic, which is tested on two sets of benchmark instances, performs better than existing approaches. Other examples of the application of simulated annealing in examination timetabling can be found in Bullnheimer (1998), Wilke and Ostler (2008), Merlot et al. (2003) etc. A tutorial on simulated annealing can be found in Burke and Kendall (2005).

2.8.5 Great Deluge Algorithm (GDA)

In 1993, Dueck introduced the great deluge algorithm (GDA) that operates in a similar way to simulated annealing (SA). However, GDA uses an upper limit (often referred to as the water level) as the boundary of acceptance, rather than a temperature. The algorithm starts with a boundary equal to the initial solution quality. It accepts worse solutions if the cost (objective value) is less than the boundary which is lowered in every iteration according to a predetermined rate (known as the decay rate). Figure 2.5 show the great deluge algorithm procedure. GDA only involves one parameter setting (decay rate) which is an advantage over SA (among others), since the effectiveness of

a meta-heuristic technique is often dependent on parameter tuning (Petrovic and Burke, 2004).

```
Choose an initial configuration
Choose the "rain speed" UP > 0
Choose the initial WATER-LEVEL > 0
    Opt: choose a new configuration which is a stochastic small
         pertubation of the old configuration
         Compute E := quality (new configuration)
         If E > WATER_LEVEL then
              old configuration := new configuration
              Water_level := water_level+ up
         If a long time no increase in quality or too many iterations
 Then stop
Goto Opt
```

Figure 2.5 Great deluge algorithm for maximisation (Dueck, 1993)

Dueck (1993) applied GDA to the travelling salesman problem. The decay rate used was the difference between the boundary and the length of the current tour divided by 500 or a fixed decay rate of 0.01. GDA was able to produce good quality solutions. Burke and Newall (2003) investigated GDA on examination timetabling problems. The decay rate is computed as the initial solution multiplied by a user provided factor divided by the number of iterations. The algorithm was run for up to 200,000,000 iterations and the search terminated if there was no improvement in the last 1,000,000 iterations. They compared the performance of the great deluge algorithm with simulated annealing and hill climbing, and concluded that GDA was superior to the other two algorithms.

Burke et al. (2004) implemented time-predefined GDA for the examination timetabling problem. The algorithm includes two user-defined parameters; (a) *computational time* (amount of time allowed) and (b) the desired solution (an estimation of the required cost value). The decay rate is calculated as the difference between the initial solution and the desired solution divided by the computational time (or number of iterations). The time-predefined GDA was able to produce good quality solutions. McMullan (2007) implemented an extended great deluge algorithm for the course timetabling problem. McMullan uses a steeper decay rate (with the decay rate propotional to 50%

of the entire run on the first stage and 25% on the remaining runs). This forces the algorithm to reach better quality solutions as early as possible. The algorithm is allowed to 'reheat' (similar to simulated annealing), which widens the boundary condition, thus allowing worse moves to be accepted. Silva and Obit (2007) use a non-linear decay rate to control the boundary and allow the boundary to rise when its value is about to converge with the current solution. Experiments on the course timetabling problem revealed that the non-linear GDA gives good quality solutions.

McCollum et al. (2009) applied the extended great deluge to the ITC2007 examination datasets using a 2-phase approach (e.g. construction and improvement). The initial solution is constructed using an adaptive ordering heuristic (Burke and Newall, 2004). Improvement is carried out using an extended great deluge algorithm that includes a reheating mechanism. The approach was able to return good solutions compared to other currently published results.

GDA has also been hybridised with other methods, Abdullah et al. (2009) hybridise GDA with TS. Their algorithm applied four neighbourhood moves (at every iteration) and selected the best solution that was generated. If there is no improvement within a specified time, the boundary is increased randomly within a value zero and three. The approach gave good results when applied to the course timetabling problem. Recently, Turabieh and Abdullah (2011) hybridised GDA with the electromagnetic-like mechanism (EM). They applied the technique to solve the Toronto dataset and the ITC2007 datasets. The EM uses an attraction–repulsion mechanism that aims to move solutions toward high quality solutions. Each candidate solution has a charge (related to the objective function value) that represents the magnitude of attraction or repulsion of the solution over the sample population. The method is able to produce good quality solutions for some of the ITC datasets.

Muller (2008) implement a search algorithm that consists of Iterative Forward Search (IFS), hill-climbing (HC) and great deluge algorithm (GDA) to the examination track of the ITC2007. The initial solution is generated using IFS, while HC and GDA are used to improve the solution. HC is used to improve the initial solution until it reaches a local optimum. Then GDA is used to further improve the solution. The multi phase

approach was implemented on the ITC2007 competition datasets and it produced the best result compared to other entrants. Muller (2008) was named winner of the first track (exam) and the third track, and it was placed fifth in the second track.

Based on the above, it is shown that GDA able to produce a good quality solution. Furthemore the algorithm is easy to understand and implement and this attracted us to explore the method. We implement GDA for solving a real world examination timetabling problem from Universiti Malaysia Pahang (UMP), by improving the result obtain from the constructive phase (graph heuristics).

2.8.6 Variable Neighbourhood Search (VNS)

The success of a meta-heuristic is determined by the technique itself and the neighbourhood structure used during the search (Ahuja, Orlin and Sharma, 2000; Thompson and Dowsland, 1998). As mentioned previously most meta-heuristic techniques are often dependent on parameter tuning (Petrovic and Burke, 2004). Many methodologies in the literature (e.g. simulated annealing and tabu search) generally use neighbourhood structure throughout the search by selecting the best result and usually focus more on the parameters that affect the acceptance of the moves rather than on the neighbourhood structures. Figure 2.6 show the variable neighbourhood search procedure.

VNS was introduced by Mladenović and Hansen (1997). It is based on the strategy of using more than one neighbourhood structure and changing them systematically during the local search. This helps VNS explore a variety of possibilities and jump to a new solution. The use of many neighbourhoods allows VNS to more effectively explore the search space (Abdullah et al., 2005; Burke et al., 2010a etc). VNS works by first determining the set of predefined neighbourhood structure k, where $k = 1,\ldots,K$ is the total number of neighbourhood structures used in the search. Let $f(s)$ be the quality of the solution s. The local search starts by randomly generating a solution s' from the kth neighbourhood. Starting from an initial solution s', the local search sequentially visits the kth neighbourhood of s' until a local optima s'' is obtained. The solution s'' is accepted if $f(s'')$ is better than $f(s)$. Whenever a neighbourhood structure generates a

better solution, the search starts over from the first neighbourhood ($k = 1$). Otherwise, the next neighbourhood is employed ($k = k + 1$).

Initialization: Select the set of neighbourhood structures N_k, for $k = 1,...,k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following sequences until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2) *Repeat* the following steps until $k = k_{max}$:

(3) *Shaking*. Generate a point x' at random from the kth neighbourhood of ($x' \in N_k(x)$);

(4) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so obtained local minimum;

(5) *Move or not*. If the local minimum x'' is better than the incumbent x, move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Figure 2.6 Variable Neighbourhood Search procedure (Hansen and Mladenovic, 2005)

Abdullah, Burke, and McCollum (2005), propose a strategy of combining VNS with a tabu list (Glover and Laguna 1993). The tabu list (set to 2) is used to hold neighbourhood structures that perform poorly and prevent them from being chosen in the next iteration, thus allowing the search to explore other possible areas of the search space. An exponential monte carlo acceptance criterion were used to enhance the exploration of the search space together with twelve neighbourhood structures. They also investigated the ordering of the neighbourhood, where an ordering strategy forces the search to return to the first neighbourhood if an improvement is found, whilst search will continue with the current neighbourhood (for non-ordering strategy). The result shows that the ordering strategy generates good results.

Burke et al. (2010a) hybridise variable neighbourhood search (VNS) with genetic algorithms. They investigate a number of different neighbourhood structures that include (a) descent–ascent that accepts worse moves (with a probability), (b) biased VNS involve moving an exam (using Kempe-chain) that causes a high penalty, (c) problem-specific neighbourhoods involve reducing the number of neighbourhoods and (d) different initialisation strategies (i.e greedy and a random construction technique). Statistically analysed results show that problems are dependent on the neighbourhoods,

where certain neighbourhoods might show an improvement on one problem but not for other problems. The proposed technique is able to produce high quality solutions on benchmark problems, however it requires a relatively large amount of computational time.

As described above, trajectory based approach focus more on the exploitation rather than on exploration (Al-Betar, Khader and Gani, 2007 and Chiarandiani et al, 2006) although they do accept non-improving moves. In the following section, we are going to describe population based approaches, which concentrate more on exploration of the search space.

2.8.7 Genetic Algorithms (GA)

Genetic Algorithms (Burke and Kendall, 2005) are a population based search which uses the principle of biological evolution to generate better solutions from one generation to another (Ross and Corne, 1995 and Burke et al., 2010a). Genetic algorithms were popularised by Holland (1975). The methodology employs operators known as genetic operators (i.e. *selection, crossover* and *mutation*) that manipulate individual solutions (also referred to as *chromosomes*) in a population for a number of generations (or iteration) in order to improve the cost value. The chromosome is represented as a string that contains the solution information. Several parameters need to be considered when applying genetic algorithm to a given problem such as population size, crossover rate, mutation rate and the number of generations (Goldberg 1989, Pham and Karaboga 2000, Burke and Kendall 2005).

Genetic Algorithm (GAs), start from an initial population of (often) random solutions. Each of these solutions is known as an individual and they each have a cost value (fitness) evaluated based on the objective function. Next, is a selection phase where the individuals will be chosen by a selection operator to undergo the recombination process. In the recombination phase, crossover and mutation operators are used to explore the solution space, thus creating new individuals. The newly created individuals replace old individuals (usually the worst individual based on their fitness).

This process is repeated until a stopping criterion is reached, which may be the maximum number of generations or a time limit. Figure 2.7 show the genetic algorithm procedure.

```
InitialisePopulation P
For each sol_i from P
    CalculateFitness (sol)
repeat
    select two parents sol_1 and sol_2 from P
    child = crossover (sol_1, sol_2)
    mutate (child)
    calculateFitness (child)
    replaceSome (p, child)
until stop condition not satisfied
```

Figure 2.7 Genetic Algorithm procedure (Cuupic, 2009)

Corne, Fang and Mellish (1993) employed a genetic algorithm for solving examination timetabling. The length of the chromosome was set as the number of examinations. The solution obtained from this algorithm was found to be better than a manual solution. In order to avoid infeasible solutions, Ross, Corne and Fang (1994) proposed using only the mutation operator to generate offspring solutions. Experimental results showed that their approach outperformed the genetic algorithm that used a uniform crossover operator. They applied a repair mechanism to overcome the infeasibilities due to the direct chromosome representation that generated infeasible offspring solutions.

Chu and Fang (1999) investigated genetic algorithms and tabu search approaches to schedule examination timetables and compared the performances of these two techniques. The investigation concentrated on the quality of the examination timetable and the time spent in producing the timetable. These experiments show that TS can produce better solutions, with less computing time than those produced by GA. However GA can produce several different near optimal solutions simultaneously because it holds a population of chromosomes which may not originate from the same parents. A grouping genetic algorithm has been applied by Erben (2001) for graph colouring and examination timetabling problems. In a grouping genetic algorithm, a

chromosome is made up from a group (of *genes*) which is different from a straightforward direct representation in a standard genetic algorithm.

Usually, the quality of a solutions produced by population-based algorithms outperformed by trajectory methods. The reason being caused by premature convergence, where population based algorithms are more concerned with exploration than exploitation (Al-Betar, Khader and Gani, 2008). Therefore much recent research has involved the hybridisation of genetic algorithms with trajectory methods to optimise the individual result.

Massoodian and Esteki (2008) implemented genetic algorithm-based approaches to solve the ITC2007 course timetabling problem (track 3). The approach consists of two stages with local search being applied (on the best chromosome) at each stage to further refine the chromosome. The first stage concentrates on finding a feasible solution, while the second stage minimises violations of the soft constraints. The approach was able to produce good solutions in less computational time compared to using GA alone.

Jat and Yang (2009) proposed a hybridisation of a genetic algorithm with local search to solve the course timetabling problem (post-enrolment) from ITC2007. The problem is solved in two phases, where in the first phase the genetic algorithm uses information from previous good individuals to guide the generation of offspring with local search techniques to improve the quality of the individuals. In the second phase, tabu search is used on the best solution obtained to try and improve the solution. The experimental results show that the proposed hybrid approach is better than, or comparable to, all other tested methods.

2.8.8 Ant Colony Optimisation (ACO)

Ant colony optimisation is a population based method proposed by Dorigo, Maniezzo and Colorni et al. (1996). ACO is inspired by the behaviour of ants, and the way they forage for food (that is, through cooperation by depositing trails of pheromone). Costa

and Hertz (1997) developed a method called ANTCOL for addressing graph colouring problems using ant colony optimisation and a sequential heuristic. In successive generations, each ant colours the vertices using static (i.e. *random, largest first, smallest last*) or dynamic (i.e. *saturation degree, recursive largest first*) constructive methods. The probability value of the pheromone is used to select the colour for each vertex. Experimental results show that the dynamic methods perform significantly better than static methods. This research highlights the promise of using ant colony optimisation in successfully solving examination timetabling problems.

Dowsland and Thompson (2005) investigated the application of ant colony optimisation for the examination timetabling problem. The objectives of their research were, firstly to compare the performance of ANTCOL on typical timetabling graphs with a set of random graphs created by Costa and Hertz (1997); secondly, the authors wished to identify promising constructive heuristic combinations, trail calculations and ANTCOL parameter values. Experimental results show that the modification of ANTCOL applied to the examination timetabling problem is competitive with the best published approaches in the literature in minimising the number of timeslots required for a feasible timetable.

Eley (2006 and 2007) implemented a Max-Min and an ANTCOL approach for the examination timetabling problem. Two algorithms were tested on the Toronto datasets using the formulation described in Carter, Laporte and Lee et al. (1996). However, they also included a clashing penalty value of 10,000 as the proposed algorithm does not guarantee a conflict free solution. Fifty ants were used with a fixed value for the evaporation rate and pheromone interval value. Different weighting factor (α and β) were tested. The results show that the approach does not generate outstanding results however its performance is comparable with other approaches.

2.8.9 Memetic Algorithms (MAs)

Genetic algorithms perform a search across the entire search space without strictly focusing on a potentially good area of the search space, which may lead to lose of useful information in a good individual (Acan and Tekol 2003). However, the

advantage of genetic algorithms is that they perform multiple directional searches using a set of candidate solutions (Gen and Cheng 1997) which can be of benefit by including a local search to refine the (best) chromosome. This is known as a memetic algorithm. Memetic algorithms represent evolutionary based approaches combined with a local search. The concept of memetics originates from Dawkins (1976) where they are described as memes that act as units of information that are passed around the society. The disadvantage is that each generation takes considerably longer, but this can be justified if sufficiently more is achieved per generation than if local search were not used. Figure 2.8 show the memetic algorithm procedure.

Burke, Newall and Weare (1996) employed a memetic algorithm for examination timetabling problem. They include a light and heavy mutation (small and large scale alteration respectively) as well as deterministic hill climbing. The aim of the work is to produce a feasible solution whilst keeping the penalty as low as possible. The method was implemented on Nottingham and Toronto datasets. Experimental results show that the method is able to produce feasible and good quality solutions (during that time). In 1999, Burke and Newall extended the above work and proposed a multi-stage memetic algorithm. The algorithm is applied to a subset of examinations while the next subset is scheduled on top of the previously scheduled events. A fixed length timetable is used to schedule the events. To avoid infeasibilities, exams are sorted according to their difficulty (i.e. largest degree, colour degree and saturation degree), together with a look ahead strategy. Experimental results show that the solution quality is better when compared to employing a memetic approach alone.

```
Create initial population
    Repeat
        8.1 Take each individual in turn:
                Choose a mutation method (light or heavy mutation)
                Apply mutation operator to chosen individual
                Apply hill-climbing to individual just created.
                Insert it into the population.
        8.2 Select a half of them to reduce the population to its original size
    Until termination condition is true
```

Figure 2.8 Memetic algorithm (Nguyen, Ta and Duong, 2005)

Nguyen, Ta and Duong (2005) present a modification of memetic algorithm presented by Burke, Newall and Weare (1996). They applied the method to solve the university examination timetabling from HoChiMinh City University of Technology. They applied the same evolutionary operators as in Burke, Newall and Weare (1996) but include penalty-based and constraint-type based Hill Climbing. Their approach is able to produce good quality solutions, although take more time. However, they claim that the modified approach is faster compared to Burke, Newall and Weare (1996). In 2004, Burke and Landa Silva identified and discussed the effective strategies when designing memetic algorithms for scheduling and timetabling problems. The suggested strategy involves dealing with infeasibility (i.e. prevent the occurrence of infeasible solutions or applying a repair mechanism), approximation of fitness evaluation using linked list data structures (to reduce the run time) and a right balance between genetic and local search methods.

Other related work on memetic algorithm include Abdullah, Turabieh and McCollum (2009) and Krasnogor and Smith (2005).

2.8.10 Hyper-Heuristics (HH)

The development of hyper-heuristics is motivated by the goal of an increased level of generality for automatically solving a range of problems (Burke, Kendall and Soubiega, 2003). Most meta-heuristics in the literature operate directly on a search space of solutions but a hyper-heuristic operates on a search space of heuristics (Burke, Petrovic and Qu, 2006). Hyper-heuristic can be categorised into two groups: heuristic *selection* and heuristic *generation.*

During the early introduction of hyper-heuristic, they could be thought of as heuristics that are able to intelligently choose a heuristic to solve a problem (Hussin, 2005; Burke et al., 2007; Qu et al., 2009 etc; Pillay and Banzhaf, 2009). This hyper-heuristic framework is provided with a set of pre-existing heuristics and the task is to discover a good sequence to effectively solve the problem indirectly. In 2003, Burke, Kendall and Soubeiga employed a tabu search as the high level heuristic to search through a space

of moving strategies for course timetabling and nurse rostering problems. The proposed approach shows good results on both of the problems considering the generality of the approach. Later in 2005, Burke, Landa Silva and Soubeiga extended the above work aiming at investigating the learning of low level heuristics that are suitable and effective for individual objectives in multiple-objective space allocation and course timetabling problems. The approach shows promising results compared with the state-of-the-art approaches.

Burke et al. (2007) implemented tabu search hyper-heuristic using graph heuristics for the educational timetabling problem. A set of *low level* heuristics (rather than solutions) represents the search space. Tabu search is used to search for the list of low level heuristics (randomly) without considering the details of the actual solutions. The heuristic sequence is used to order the events (courses or exams) that are not scheduled yet at that iteration. They include a 'failed list', scheduling more than one exam/course at each step and SD as an initial heuristics list to speed-up the run time. The results are within the range of the best results reported in the literature. They also claim that when being employed on its own, SD performs the best in most cases due to its ability to *dynamically* order the events according to the number of remaining valid timeslots.

Qu and Burke (2009) extends the work from Burke et al. (2007) by proposing an adaptive approach (rather than tabu search) where heuristics are dynamically hybridised during solution construction. The other heuristics (LD, LWD and LE) are randomly hybridised into the list of SD. They conclude that the adaptive approach able to produce comparable result (especially hybridisation with LWD) with the current best approaches in the literature. They claim that the adaptive hybrid approach is an efficient and much simpler method compared to Burke et al. (2007) which required much more computational time. Qu, Burke and McCollum (2009) also investigate interative approach that hybridised graph heuristics adaptively.

Other related research in hyper-heuritics includes by Han and Kendall (2003) that uses genetic algorithm hyper-heuristics and Kendall and Hussin (2005) that applied tabu search hyper-heuristics, etc.

2.9 Conclusions

This chapter has introduced the University timetabling problem with a focus on the examination timetabling problem. The examination problem varies in their constraints from one institution to another. From the literature, the most commonly used datasets are from Toronto, Nottingham and Melbourne. There are some others from real world exam (e.g. UKM, UiTM and ITC2007 (which is gaining in popularity)). Based on the constraints of these datasets, we notice that there is a gap in terms of the range of constraints compare to the UMP examination dataset, which we study is this thesis.

Various methodologies (e.g. heuristics, meta-heuristics and hyper-heuristics) especially meta-heuristics have been applied to solve the benchmark examination timetabling problem. However, the success of meta-heuristics is dependent on parameter tuning (Petrovic and Burke, 2004) which would be a problem for non-experts (e.g. a timetable officer). Therefore we propose a method that is simple and has parameter(s) which are easy to set. Further discussions on the propose method will follow in the following chapter.

Chapter 3

A Case Study of the UMP Examination Timetabling Problem

This chapter comprises five sections. Section 3.1 presents an introduction to the Universiti Malaysia Pahang (UMP). Section 3.2 describes the UMP examination timetabling process. Section 3.3 shows the UMP examination timetabling constraints, listing the UMP examination constraints and the UMP invigilator constraints. The dataset used throughout the work is discussed in section 3.4. Lastly in section 3.5 we present our conclusions.

3.1 Universiti Malaysia Pahang (UMP)

The Universiti Malaysia Pahang (UMP), formerly known as Kolej Univerisiti Kejuruteraan dan Teknologi Malaysia (KUKTEM), was established in 2002 and is located in Pahang, Malaysia. In 2007, UMP consisted of five faculties with a total of 3,550 students. The faculties are the Faculty of Mechanical Engineering (FKM), the Faculty of Computer Science & Software Engineering (FSKKP), the Faculty of Chemical & Natural Resources Engineering (FKKSA), the Faculty of Electrical & Electronics Engineering (FKEE) and the Faculty of Civil & Environmental Engineering (FKASA). Currently, a total of 17 programs are being offered by these faculties which include two types of certificates; Diploma and Bachelor degree.

However, in 200809 the total programs offered increased to a total of 23 programs. This is because of the establishment of new faculties; Faculty of Industrial Sciences & Technology (FIST) and Faculty of Manufacturing Engineering & Technology Management (FKPPT) with one and four programs respectively. Additionally, one new program offered by FKASA. This has resulted in an increase in the total number students to 4284. As a new University, a good decision making system is important to aid University operations. UMP is currently situated in a temporary campus, which presents many challenges in terms of available space, logistics and the human resources in order to manage the process. In addition to these limitations, the UMP examination timetabling problem has other challenging constraints which have never been tackled before in the literature (at least as far as we are aware).

In UMP, the Academic Management Office is responsible for planning and managing the entire academic process. It provides all the academic space and facilitates academic affairs. All this is done with the aid of an Information Management System (IMS). This system encompasses a complete student life cycle process; from student intake to graduation. One of the modules in the IMS includes generating an examination timetable which has been used since 2003. However, although this proprietary system has been successful in producing the examination timetable as it involves manual processes in order to achieve a feasible solution. Moreover, the proprietary system is unable to determine the quality of the solutions it produces due to having no underlying mathematical model (that we are aware of) that allows us to calculate the effectiveness of the generated timetable. Therefore, one of our research objectives is to develop a formal model for the UMP *examination-timeslot-room* assignment and the *invigilator* assignment in order to evaluate the effectiveness of the solution produced by the proprietary system and thus enable a comparison with other methods.

3.2 UMP examination timetabling process

Generating the examination timetable involves several processes as well as interaction between students, administration and lecturers. The UMP examination timetabling processes are as follows:

1) The process starts with the lecturer providing information and their requirements on the exam papers. The information includes *combined* exam papers (*combined* exams refer to several exams which need to be scheduled at the same time), course codes which do not require a final examination etc.
2) Next, administrators will generate an examination data report from step 1 for the lecturer or faculties to verify the information received.
3) Then, a draft timetable is prepared and distributed to students and faculties for corrections or amendments. The first draft includes assigning exam papers to timeslots, to rooms and scheduling the invigilators. This phase normally takes a couple of iterations before a final timetable is published.
4) The timetable is then updated based on feedback received. The final timetable is published to students and faculties.

The process described above takes place every semester, because, every semester each student registers for a different set of courses. Hence, the exam timetable for each semester is only valid for that particular semester. Therefore, in practice the exam timetable process normally starts a few months before the actual examination period.

3.3 UMP examination timetabling constraints

Apart from an increasing number of students and programs offered, the UMP *examination-timeslot-room* assignment problem has other challenging constraints which have never been tackled before in the literature, at least, as far as the author is aware. These constraints are the room distance for an exam in multiple rooms and the splitting of an exam across several rooms. The room distance requires that when an exam is being split across different rooms, the rooms should be as close as possible to one another and the rooms must be in the same building. This is to provide the lecturers easier, and quick, access to the examination rooms to answer any queries during the examination. The constraints are listed in section 3.3.1 and the invigilator assignment constraints are listed in section 3.3.2.

3.3.1 UMP examination constraints

Below are the constraints for the UMP examination timetabling problem. The hard constraints for the UMP examination timetable problem are as follows:

H_E1: No student should be required to sit two examinations simultaneously.

H_E2: The total number of students assigned to a particular room(s) must be less than the total room capacity.

H_E3: Only one examination paper is scheduled to a particular room. That is, there is no sharing of rooms with other exam papers (even if enough seats are available to fit in another exam). However, some exams can be combined with others for the following reasons:

- The same examination for different academic programs.
- Lecturers request exam paper to be combined. In this case, the lecturer might teach different courses but with similar content.
- Faculties request that exams are combined. The combined exam papers contain similar (or almost similar) exam questions.

The request for combined exams is done before the exam schedule is generated. For the combined exams, we give a new examination code and treat it as one large exam.

H_E4: The size of each exam room in UMP is relatively small (less than 100) and with a large number of registered students for each exam, this inevitably leads to splitting exams into different rooms. In splitting the exam into different rooms we need to allocate the rooms as close as possible to each other (this actually represents a soft constraint, see below) and the rooms MUST be in the same building (a hard constraint).

In measuring the quality of the solution, the soft constraints are as follows:

S_E1: Each set of student examinations should be spread as evenly as possible over the exam period.

S_E2: The distance between exams room for the same exam should be as close as possible to each other (and within the same building, see H_E4).

S_E3: There is a penalty associated with splitting an exam across several rooms, as we would like an exam to be in a single room whenever possible.

These constraints are different from the benchmark datasets (see chapter 2). Having these constraints complicates the problem. As reported in the literature, the capacitated problem is more difficult to solve compared to un-capacitated problem and it more closely resembles the real world (Merlot et al., 2003). A summary of the UMP exam constraints and comparison with other datasets is shown in table 3.1.

3.3.2 UMP invigilator constraints

The constraints for the UMP *invigilator-scheduling* problem are as follows:

H_I1: Invigilators or chief invigilators cannot invigilate their own exam paper. This because they need to be on standby during their exam paper to assist students with any queries.

H_I2: Chief invigilators must be a lecturer selected from the staff list. With extra tasks and responsibility for the chief invigilator, university policies only allow staff with lecturer status to be assigned as a chief invigilator.

H_I3: Staffs are not assigned to more than one invigilation duty is one timeslot.

H_I4: Staffs can only invigilate a maximum of three examinations within the exam period.

H_I5: Each room should be assigned the required number of invigilators (including chief invigilator).

In measuring the quality of the solution, the soft constraints are as follows:

S_I1: The chief invigilator duties should be evenly spread among the lecturers.

S_I2: The invigilation duty (invigilator and chief invigilator) should be evenly spread among all the staffs

Table 3.1 Summary of datasets

	Constraints	Toronto	Nottingham	Melbourne	UKM	UiTM	ITC2007	UMP
Examinations	Clash free	Hard	Hard	Hard	Hard	Hard	Hard	Hard
	Scheduled all exams		Soft	Soft	Hard	Hard		Hard
	Weekend scheduled					Soft		
	Exam preference - *specified arrangement: sa* - *specified room: sr* - *large exam schedule first: lf* - *restriction on exam in particular timeslot: rt* - *scheduled combined exam in the same timeslot: ct*			Hard *(rt)*		Hard *(ct)*	Hard *(sa)* Soft *(lf)*	
	Consecutive exam - *two exam in a row: 2r* - *two exam in a day: 2d* - *two exam in a row overnight: 2n* - *three exam in a day: 3d*		Soft *(2d & 2n)*	Soft *(2d & 2n)*	Hard *(3d)* Soft *(2r)*		Soft *(2r and 2d)*	
Timeslot related	Timeslot preference - *minimise/avoid usage: tu*						Soft *(tu)*	
	Timeslot length - *mixed duration of exams in one timeslot: mt*						Hard Soft *(mt)*	
	Spreading - *specified spread: ss*	Soft	Hard *(ss)*	Soft	Soft	Soft	Soft *(ss)*	Soft
Rooms related	Room distance							Soft
	No sharing of room with other exams - *for specified exam only: se*				Hard *(se)*			Hard
	Room preference - *consecutive exam scheduled in the same room: cr* - *minimise/avoid usage: ru* - *specified room: sr*				Hard *(cr)*		Hard *(sr)* Soft *(ru)*	
	Split exam into different rooms - *same building only: sb* - *as close as possible: cp*							Hard *(sb)* Soft *(cp)*
	Capacity - *total seats: ts* - *individual room: ir*		Hard *(ts)*	Hard *(ts)*	Hard *(ts and ir)*		Hard *(ir)*	Hard *(ir)*

Hard = hard constraint; Soft = Soft constraint; shaded cell = constraint not considered.

3.4 Datasets

The investigations were carried out using two different datasets from semester 1-2007/08 and semester1-200809. Table 3.2 summaries the datasets.

3.4.1 Semester1-200708

The total number of examination papers is 252, across 17 programs offered by 5 faculties. However, due to the combined exams requirement, the dataset has been pre-processed and the combined exams are given a new examination code and treated as one large exam. This results in a total number of 157 examinations. The total number of students is 3,550 with 12,731 enrolments. The conflict matrix density is 0.05, which means that 5% of students are in conflict among the examination papers. The number of exam days and timeslots are 10 and 20 respectively. There are only two timeslots on each examination day. The total available exam space for this dataset is 24 rooms, with each room having a given capacity. The number of staff available for the invigilation duty is 227 staff. From the 227 staff, 152 are academic staff and 75 are non-academic staff. Each room requires 2 invigilators (including a chief invigilator). 169 lecturers are involved in teaching the 157 exams. The 169 lecturers are not necessarily included in the staff list for invigilation duty.

Table 3.2 Summary of UMP investigated datasets

Categories	Semester1-200708	Semester1-200809
Exams	157	165
Students	3,550	4,284
Enrolments	12,731	15,416
Conflict density	0.05 (5%)	0.05 (5%)
Timeslot per day	2	2
Rooms	24	28
Invigilator	152[a] 75[b]	207[a] 125[b]

[a] *number of lecturers,* [b] *number of non-lecturers*

3.4.2 Semester1-200809

The total number of examination papers is 193 across 23 programs offered by 7 faculties. Due to combined exams request (including '*relax*' exam by the timetable officer) the total number of exams is 165. The total number of students is 4284 with 15,416 enrolments. The conflict matrix density is 0.05, which means that 5% of students are in conflict among the examinations paper. The number of staff available for invigilation duty is 332 staff. From the 332 staff, 207 are academics and 125 are non-academics. The total rooms allocated for this dataset are 28 rooms with each room requiring a minimum of two and a maximum of four invigilators (including a chief invigilator). 194 lecturers teach the courses for the 165 exams. The number of exam days and timeslots are 10 and 20 respectively. There are two timeslots on each examination day. The total available exam space for this dataset is 28 rooms, with each room having a given capacity.

3.5 Conclusions

This chapter has presented the UMP examination timetabling problem. A description of the UMP examination and invigilator constraints was presented. The UMP examination timetabling problem contains additional constraints which consider individual room capacities, whilst not allowing rooms to be shared by multiple exams (unless exams are combined, where they are treated as one exam). In addition, UMP also has a distance penalty cost used when an exam is split across more than one room and a splitting penalty cost as it is preferable to use only one room for a given exam. Having the individual room capacities and prohibiting having more than one exam in a room constraint, we believe that it is best to solve the examination assignment problem sequentially as an *exam-timeslot-room assignment*. After that, the invigilation problem is solve after the *exam-timeslot-room* assignment. The UMP uses its own staff to invigilate exams, and this lead to many invigilation constraints.

Two datasets from different semester have been collected for experimental purposes. The dataset has been pre-processed where the combined exams are treated as one large exam and given a new examination code.

The next chapter describes the mathematical model of the UMP examination timetabling problem and a constructive heuristic used in generating the solution. Our constructive heuristic is able to produce a better solution compare to the timetable produced by the UMP proprietary software.

Chapter 4

The Examination Timetabling Problem at Universiti Malaysia Pahang: Comparison of a Constructive Heuristic with an Existing Software Solution

The work presented in this chapter was published in the European Journal of Operational Research (Kahar and Kendall, 2010a). This work presents a real world, capacitated examination timetabling problem from Universiti Malaysia Pahang (UMP), Malaysia. The problem has constraints which have not been modelled before, these being the distance between examination rooms and splitting exams across several rooms. These constraints provide additional challenges in defining a suitable model and in developing a constructive heuristic. One of the contributions of this work is to formally define this real world problem. A further contribution is the constructive heuristic that is able to produce good quality solutions for the problem, which are superior to the solutions that are produced using the university's current software. Moreover, our method adheres to all hard constraints which the current systems fails to do.

Section 4.1 begins with an introduction to the motivation on solving the UMP examination timetabling problem. This dataset has several new constraints in addition to those commonly used. A formal model of the problem is presented in section 4.2. In section 4.3, we describe the experimental setup for our proposed constructive heuristic.

In section 4.4, a comparison between the solutions achieved with the current method employed by Universiti Malaysia Pahang (which is produced using a proprietary system), and our method, is presented in order to evaluate the effectiveness of the proposed methodology. In section 4.5 and 4.6, we present the contribution and conclusions respectively.

4.1 Introduction

The capacitated examination timetabling problem considered room capacities along with other commonly used hard constraints in scheduling the exams. Many work in the literature investigated the un-capacitated examination timetabling problem which we believe does not describe the full aspect of the problem (McCollum, 2007; Carter and Laporte, 1996; and Qu et al. 2009). Based on the datasets described in table 3.1 and the other constraints listed in the literature (Burke et al., 1996; Qu et al., 2009), we note that there is a gap in terms of the examination timetabling datasets from the literature and many of the requirements faced by many institutions. The UMP examination timetabling problem contains additional constraints which consider individual room capacities, whilst not allowing rooms to be shared by multiple exams (unless exams are combined, where they are treated as one exam). In addition, UMP also has a distance penalty cost (applied when an exam is split across more than one room for a given exam) and a splitting penalty cost (as it is favorable to use only one room) for a given exam. A further discussion on the UMP examination timetabling problem is presented in the next section.

The solution approaches seen in literature for the exam timetabling problem can be separated into *exam-timeslot assignment* and *exam-room assignment*. The most published work seen in the literature is the exam-timeslot assignment. Only a few works discuss *exam-room assignment* (Dammak, Elloumi and Kamoun, 2006). Both the un-capacitated and capacitated (as total seating capacity) problem (i.e. benchmark dataset) can be solved using a two-phase approach, as both allow more than one exam in an examination room. This will provide a feasible solution in the *exam-room assignment* phase as long as the capacity of rooms is greater than the number of

students (Dammak, Elloumi and Kamoun, 2006). However, if individual room capacities are used, as well as prohibits sharing of classroom among the exams might not guarantee that we are able to find a feasible solution through the two-phase approach. We might even need to introduce a solution repair mechanism in order to arrive at a feasible solution. Therefore, in this problem, we are going to solve the UMP examination timetabling problem sequentially as an *exam-timeslot-room* assignment.

4.2 Problem formulation

In this section, we present the formal model of the UMP examination timetabling problem as discussed in chapter 3.

Indices

i, j $1...N$

r, p $1...R$

s $1...S$

t $1...T$

Parameters

N The number of examinations

R The number of examination rooms

S The number of students

T The number of available timeslots

S_i The number of registered students in exam i

R_t The number of examination rooms available at timeslot t

B_r The building for room r

f_r The total capacity for room r

c_{ij} The conflict matrix where each element $(c_{ij}, i, j \in \{1...N\})$ is the number of students that have to take both exam i and j. The conflict matrix is a symmetrical matrix of size N, where diagonal elements $c_{ii} = S_i$

d_{rp} The distance matrix where each element (denoted by $d_{rp}, r, p \in \{1 \ldots R\}$) is the distance between rooms r and p. The distance matrix is a symmetrical matrix of size R, where diagonal elements $d_{rr} = 0$

Decision variables

x_{it} 1 if examination i is assigned to timeslot t, 0 otherwise

y_{ir} 1 if examination i is assigned to room r, 0 otherwise

z_{rt} 1 if room r is assigned to timeslot t, 0 otherwise

The objective is to spread out examinations over the exam period (timeslots) for each student, minimise the distance between rooms of an exam that is being held in multiple rooms and to minimise splitting an exam over several rooms. Therefore our formulation is as follows:

$$\text{(Minimise) } F(x) = F_1 + F_2 + F_3 \qquad \text{(Eq.1)}$$

The first component of the cost, F_1 (spreading exams over the exam period, S_E1) is shown in Eq.2.

$$F_1 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij} \cdot proximity(t_i, t_j)}{2S} \qquad \text{(Eq.2)}$$

and

$$proximity(t_i, t_j) = \begin{cases} 32/2^{|t_i - t_j|} & if\ 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases} \qquad \text{(Eq.3)}$$

Where t_i and t_j specifies the assigned timeslot for examination i and j (i, j $\in \{1, \ldots, N\}$). Eq.2 represents the cost for an exam i that is given by the proximity value multiplied by the number of students in conflict. Proximity values of 16, 8, 4, 2 and 1 are used here. For example, if a student has two consecutive examinations then a proximity value of 16 is assigned. If a student has two examinations, with a free timeslot in between, then a value of 8 is assigned. Two empty periods correspond to a penalty of 4 and so on. These proximity values were introduced by Carter, Laporte and

Lee (1996) and have been widely used by other researchers (see Burke et al., 2004; Ayob, Abdullah and Malik, 2007; Abdullah, 2006).

The second component of the cost, F_2 (distance of an exam in multiple rooms, S_E2) is shown in Eq.4:

$$F_2 = \frac{\sum_{i=1}^{N}\sum_{r=1}^{R-1}\sum_{p=r+1}^{R} d_{rp} y_{ir} y_{ip}}{N} \qquad \text{(Eq.4)}$$

Eq.4 represents a cost for an exam i that is scheduled in multiple rooms. A subset of the distance matrix is shown in figure 4.2.

The third component of the cost, F_3 (splitting exam, S_E3) is shown in Eq.5:

$$F_3 = \frac{\sum_{i=1}^{N} m_i - 1}{N} \qquad \text{(Eq.5)}$$

Where m_i is the number of rooms exam i has been split across. It can be calculated using the following formulation, $m_i = \sum_{r=1}^{R} y_{ir} \; \forall i \in \{1,...,N\}$. Eq.5 represents a cost for an exam i that is being penalised for splitting the exam in multiple room $(m_i > 1)$. For example, if an exam is being split into 2 rooms, then a value of 1 is given as the penalty value. Splitting the exam across 3 rooms corresponds to a penalty of 2 and so on.

Eq.1 is subject to the following constraints:

a) No student can sit two exams concurrently (clash-free requirement, H_E1). If examination i and j are scheduled in timeslot t, the number of students sitting both examination i and j must be equal to zero, i.e. $c_{ij} = 0$. This hard constraint is expressed in Eq.6:

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T} x_{it} x_{jt} c_{ij} = 0 \qquad \text{(Eq.6)}$$

b) All exams must be scheduled and each exam must be scheduled only once in available timeslots, T (see Eq.7)

$$\sum_{t=1}^{T} x_{it} = 1 \quad \text{For all } i \in \{1,\ldots, N\} \qquad \text{(Eq.7)}$$

c) Only one examination paper is scheduled to a particular room in a particular timeslot, H_E3. There is no sharing of rooms with other exam papers (even though seats might be available to fit in another exam), except for requested combined exams, which has been carried out as a pre-process operation (see Eq.8).

$$\sum_{i=1}^{N} x_{it} y_{ir} = z_{rt} \quad \text{For all } t \in \{1,\ldots,T\} \text{ and for all } r \in \{1,\ldots, R\} \qquad \text{(Eq.8)}$$

d) Exam can only be split across several rooms in the same building, H_E4 (see Eq.9).

$$\sum_{r=1}^{R-1} \sum_{p=r+1}^{R} y_{ir} y_{ip} b_{rp} = \frac{m_i(m_i - 1)}{2} \quad \text{For all } i \in \{1,\ldots, N\} \qquad \text{(Eq.9)}$$

Where

$$b_{rp} = \begin{cases} 1 & if(B_r = B_p) \\ 0 & otherwise \end{cases}$$

e) For each timeslot t, the number of rooms assigned to a particular timeslot must not exceed the maximum number of rooms available in a timeslot, R_t (see Eq.10)

$$\sum_{r=1}^{R} z_{rt} \leq R_t \quad \text{for all } t \in \{1,\ldots,T\} \qquad \text{(Eq.10)}$$

f) The total number of students assigned to a particular exam room(s) must be less than the total room capacity (see Eq.11).

$$S_i \leq \sum_{r=1}^{R} y_{ir} f_r \quad \text{For all } i \in \{1,\ldots, N\} \qquad \text{(Eq.11)}$$

4.3 Experimental setup

In this section we present our proposed constructive heuristic, along with other algorithmic details to aid reproducibility. The dataset is taken from Universiti Malaysia Pahang (UMP) for semester1-200708. The total number of examination papers is 252, across the 17 programs offered by 5 faculties (see chapter 3 for further details). The number of exam days and timeslots are 10 and 20 respectively. There are only 2 timeslots on each examination day. There are no exams during the weekend (Saturday and Sunday). We capture this by introducing gaps in our timeslots indices. Therefore the timeslots can be represented as shown in figure 4.1. In figure 4.1, timeslot 1 and 2 refer to day 1, timeslot 3 and 4 refer to day 2 etc. Notice that indices 11 to 14 are missing. This is because those indices refer to Saturday and Sunday.

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24)

Figure 4.1 Timeslot indices

The total available exam space for this dataset is 24 rooms, with each room having a given capacity. To assist our constructive heuristic in the process of searching for the most suitable room(s) and minimising the room related cost value, we generate a list of room groupings (based on the list of rooms provided). These pre-determined room groupings are generated within the same building only. Note that we limit the room groupings up to a maximum of 4 possible rooms for each exam. In our observations, 4 rooms are adequate to satisfy any exam capacity. Besides, increasing the room grouping possibilities (>4) will increase the distance cost, splitting cost and the search space. The room groupings are sorted in decreasing order based on the total room(s) capacity. By doing so we could directly search for suitable room(s) and end the search procedure when an unsuitable room capacity is encountered.

To illustrate the procedure we provide the following example. Assume, we have 5 rooms in 2 different buildings, where 4 of the rooms are in the same building, and each room has a specific capacity (see figure 4.2). The travel cost for rooms in different buildings is not shown, as this is not permitted. Therefore, we could create 15 room groupings with 14 room groupings from building W and 1 room grouping from

building X. Referring to figure 4.3, each of the room groupings have their new total capacity, distance cost (total of the distance value prior to the distance matrix for every rooms) and splitting cost (m_i -1). These room groupings are sorted based on their capacity. Having the decreasing order of pre-determined room groupings assists the search algorithm in selecting the most suitable rooms, aiming to minimise the room related cost value and speeding up the search by stopping the room search procedure if an unsuitable room grouping capacity is encountered.

Room	Capacity	Building	WDK26	WDK28	WDK29	WDK30	XDK04
WDK26	92	W	0	2	3	4	-
WDK28	90	W	2	0	1	2	-
WDK29	40	W	3	1	0	1	-
WDK30	40	W	4	2	1	0	-
XDK04	47	X	-	-	-	-	-

Figure 4.2 Room information and distance matrix

No.	Room Grouping	Room Grouping Capacity	Distance Cost	Split Cost
1	WDK26 - WDK28 - WDK29 - WDK30	262	13	3
2	WDK26 - WDK28 - WDK29	222	6	2
3	WDK26 - WDK28 - WDK30	222	8	2
4	WDK26 - WDK28	182	2	1
5	WDK28 - WDK29 - WDK30	170	4	2
6	WDK26 - WDK29	132	3	1
7	WDK26 - WDK30	132	4	1
8	WDK28 - WDK29	130	1	1
9	WDK28 - WDK30	130	2	1
10	WDK26	92	0	0
11	WDK28	90	0	0
12	WDK29 - WDK30	80	1	1
13	XDK04	47	0	0
14	WDK29	40	0	0
15	WDK30	40	0	0

Figure 4.3 Decreasing order of pre-determined rooms grouping

Algorithm Parameters:

- $i = 1...N$ where N is the number of examinations
- $g = 1...G$ where G is the number of pre-determined *roomGrouping*
- $r = 1...R$ where R is the number of rooms,
- $c = 1...C$ where C is the candidate list size,
- $t = 1...T$ where T is the number of timeslot
- *totalSeatAvailable*(t) is the total seating capacity available calculated in timeslot t
- *capacity*(i) is the size of examination i
- *spreadCost*[c] store the spreading penalty for candidates list c
- distCost[c] store the room distance penalty for candidates list c
- *splitCost*[c] store the splitting room penalty for candidates list c
- *roomCapacity*(g) is the total room seating capacity in g,
- *distPenalty* is the room distance penalty in g,
- *splitPenalty* is the splitting room penalty in g,
- $x_{it} = 1$ if examination i is assigned to timeslot t, 0 otherwise,
- $y_{ir} = 1$ if examination i is assigned to room r, 0 otherwise,
- $z_{rt} = 1$ if room r is assigned to timeslot t, 0 otherwise,

```
1  Step 1: Ordering:
2     Sort examination N based on the Graph Colouring heuristic;
3     Sort roomGrouping G in decreasing order based on the total capacity;
4  Step 2: Assigning exams to timeslot and room(s):
5     Set i ← 1;
6     Until i = N, repeat:
7     (2.1) Set c ← 1;
8     (2.2) Until c = C, repeat:
9        (2.2.1) Set count ← 0, g ← 1 and t ← -1;
10       (2.2.2) Until t = -1 && count < 3, repeat:
11                (a) Generate t randomly and no clashing with other exams
12                (b) If t is not equal with t previously generated and capacity(i) ≤ totalseatAvailable(t), then
13                   calculate spreading penalty as spreadCost[c]
14                   Otherwise, Set t ← -1 and increase count;
15       (2.2.3) Set distCost[c] ← +∞ and splitCost[c] ← +∞;
16       (2.2.4) Do the following if t ≠ -1:
17                (a) Until capacity(i) ≤ roomCapacity(g), repeat:
18                   (i) If room g is available and distCost[c]+splitCost[c] > distPenalty(g)+splitPenalty(g), then
19                      set distCost[c] ← distPenalty(g), splitCost[c] ← splitPenalty(g)
20                   (ii) Increase g
21       (2.2.5) Increase c;
22    (2.3) Select the minimum total cost value from C and set x_it←1 y_ir←1 and z_rt←1, if t ≠ -1 for every c
23    (2.4) Increase i
24 Step 3: Verification
25    Check the solution prior to the constraints
26    Calculate the cost value
```

Figure 4.4 Pseudo-code for the examination timetable

The experiments are conducted using graph heuristic approaches including Largest Degree (LD), Largest Weighted Degree (LWD), Saturation Degree (SD) and Largest Enrolment (LE) (Carter et al., 1996). The description of these methods is presented below:

- Largest degree (LD): this heuristic takes the exams that have the most conflicts with other exams and schedules them first.
- Largest weighted degree (LWD): this heuristic is similar to largest degree except that it takes exams that have the most number of students who are involved in the conflict and schedules them first.
- Largest enrolment (LE): this heuristic takes exams with the largest number of registered students and schedules them first.
- Saturation degree (SD): this heuristic chooses exams which have the least number of available periods in the timetable that can be selected and schedules them first.

In general, the algorithm (see figure 4.4) starts (line 2) by sorting the examinations based on a graph colouring heuristic (e.g. LD, SD, etc) and also sorting the room groupings G in decreasing order based on total room(s) capacity (line 3). For all examination i, (step 2) we randomly select a timeslot t (the number of timeslots we consider is referred to as a candidate list, and we show the effect of different candidate list sizes in the results section), which is clash free and we only accept t if it is not equal with any t previously generated in C and the total available seating capacities in timeslot t (*totalSeatAvailable(t)*) able to accommodate exam i (*capacity(i)*) (line 11-12). If the total available seating capacities in t is greater or equal to exam i (*capacity(i)* $\leq$ *totalseatAvailable*(t)), we will continue to calculate the spreading penalty based on the selected timeslot and store it in *spreadCost[c]* (line 12-13). The *spreadCost[c]* value will be used later in selecting the timeslot and room with the minimum cost values (see line 22). However, if the total available seating capacities unable to accommodate the exam, the search will continue to look for other t until a number of *count* trials (line 14). Here we set a maximum of 3 trials. If after a number of *count* trials the algorithm still could not find a feasible t, then the search will proceed with the next c (line 21). Otherwise, it will continue with the room assignment which goes through the room groupings G (line 16). Selections of g is based on it capacity. If room grouping g able

to accommodate exam *i*, an availability check on the individual room(s) in the *g* is carried out and if the exam *i* can be accommodated, the room distance and the splitting penalty within the room(s) in room grouping *g* is calculated as *distPenalty(g)* and *splitPenalty(g)*) respectively (line 17-19). These values are compared with the distance cost (*distCost[c]*) and the splitting cost (*splitCost[c]*) in *c*. The value of these arrays, *distCost[c]* and *splitCost[c],* are overwritten if the distance and splitting costs are minimum. Otherwise, we will continue to search for other rooms in *G*. Once, room(s) in *g* been selected, we select the minimum cost value found by comparing each of the *spreadCost, distCost* and *splitCost* in *C* and set the decision variable to 1 (line 22). The algorithm will continue the search for all exam *i* (line 23). Lastly, we verified the solution by checking the solution to ascertain that the timeslot and rooms found satisfied the constraints and calculate the cost value (line 25-26).

4.3.1 Discarding moves sub-algorithms

The algorithm is able to find superior solutions, compared to the proprietary software, in a small amount of computational time. This is done by discarding unnecessary moves as early as possible in the algorithm. Referring to the algorithm (figure 4.4), the discarding move algorithms are as follows, and we present them here to assist in reproducibility:

a) Total available seating capacities in timeslot *t* (lines 12-14).

Line 12-14 checks the room availability prior to timeslot *t* is generated. It calculates the total available seats in *t*. If the total available seats are unable to accommodate exam *i,* (see line 14), then a new clash free timeslot *t* is generated. Having to calculate the total available seating capacities would avoid the search from selecting an inappropriate timeslot. It is good to recognize that we don't have enough room early in the search, rather than at the end, in order to make effective use of the computational time available.

b) Room grouping capacity checking (line 17).

In line 18 the algorithm will check whether the room grouping g (start at $g = 1$) able to accommodate exam i. If the condition is TRUE, the algorithm will continue to determine whether each room in g (line 18) is available or otherwise it will look for other g in the list. The room grouping search will stop once an unsuitable room grouping capacity is found (as we have already sorted the room grouping in descending order) as it will only consume computational time if the search in the G is continue.

c) Determine room availability in g (lines 18).

Lines 18, checks every room(s) in the room grouping g to determine whether the room is available or not. This is done by checking z_{rt} ($z_{rt} = 1$ if room r is assigned to timeslot t, 0 otherwise). If $z_{rt} = 0$ it means that the room is available and the search will continue to check other rooms. However, if $z_{rt} = 1$ which means that the room is unavailable, the algorithm will stop searching the room members in the selected room groupings g and continue to select the next suitable room groupings g.

d) Selecting minimum value of distance and splitting cost (lines 18).

In line 18, the algorithm will only proceed if all the rooms in room grouping g are available. Hence, it will compare the distance (*distPenalty(g)*) and splitting penalty (*splitPenalty(g)*) in g with the *distCost[c]* and *splitCost[c]*. If these penalty values are less than the current value stored in *distCost[c]* and *splitCost*[c], we will store this value in *distCost[c]* and *splitCost[c]*.

All of these discarding moves help in finding a feasible solution with minimum cost value compare to UMP proprietary software in a small computational time. In the next section we present our results.

4.4 Results

In this section, we compare the examination timetable generated by the proprietary software and the result from our proposed algorithm, shown in figure 4.4.

4.4.1 UMP proprietary software

In the solution generated by the proprietary software for semester1-200708, prior to the model being developed presented in section 4, the solution exhibited the following characteristics:

- Based on the five hard constraints stipulated by UMP, the examination timetable that was produced complied with all the constraints except for the *no student should be required to sit two examinations simultaneously* constraint (H_E1). Eight students were scheduled to sit exams at the same time and UMP had to quarantine these students.
- As mentioned previously, the quality of the solution is measured based on three objectives. The calculated cost for each of the objectives is F_1 = 8.82 for the spreading of exams (S_E1) over the examination period, F_2 = 3.63 for the distance of an exam in multiple rooms (S_E2) and F_3 = 0.71 for the number of room(s) an exam being split across (S_E3). The sum of the cost is therefore 13.16. Recall that this includes violation of the hard constraint on the clash free requirement (H_E1).

4.4.2 Graph colouring heuristic

Using the proposed heuristic, several experiments have been run with different candidate lists. In the context of this work a candidate list is how many timeslots are considered when placing an examination. Each experiment was run 50 times in order to produce average and standard deviation statistics. Every one of the 50 runs produced a feasible solution. The experiments were run on a Pentium core2 processor. The

running time for a candidate list of one is around 99 seconds and 470 seconds for a candidate list of five.

With a candidate list of one ($C = 1$), the algorithm searches randomly for one available timeslot and selects the room grouping that produces the minimum cost value for room distance and the number of splitting rooms. Referring to table 4.1, the result using a candidate list of one produces comparable solutions with the proprietary software while adhering to all the constraints. On average, our approach produces a cost value that is higher compared to the proprietary software solution (see table 4.1). However, our solutions adhere to all of the constraints compared to proprietary software, which does not. Referring to table 4.1 (column *min*), we are able to produce a solution that is 17% (13.16 compared with 10.98 ((13.16 – 10.98)/13.16 x 100%)) better when compared to the solution produced by the proprietary software. Of the heuristics we have used, largest enrolment (LE) produced the best cost value of 10.98 where the spreading cost is $F_1 = 9.01$, the distance cost is $F_2 = 1.39$ and the splitting cost is $F_3 = 0.58$ with a standard deviation of 2.10. LWD is second best with a minimum cost of 11.43 followed by saturation degree-LE, saturation degree-LD, largest degree (LD) and Saturation degree-LWD. Overall, using a candidate list of one is able to produce a good solution, which adheres to all the hard constraints (unlike the proprietary software).

When using a candidate list of five, the algorithm randomly searches for five available timeslots. For each of the timeslots selected, the algorithm will search the room groupings that give the minimum cost value (distance and splitting cost). Finally, among all the selected timeslot and room(s), we will select the one which produces the minimum total cost value. Referring to table 4.1, the result constructed using a candidate list of five produced a solution that is between 15% (13.16 compared with 11.12 ((13.16 - 11.12/13.16 x 100%)) to 64% (13.16 compared with 4.74 ((13.16 - 4.74)/13.16 x 100%)) better when compared to the proprietary software. Largest Enrollment (LE), again produces the minimum cost value (4.74). Other heuristics perform relatively the same, with respect to their ordering based on their performance, with a candidate list of one. However, with candidate lists of five all heuristics outperform the UMP proprietary software with the minimum spreading cost found

being $F_1 = 3.31$, distance cost $F_2 = 0.98$ and splitting cost $F_3 = 0.45$ with a total of 4.74 (produced using LE).

Our proposed algorithm always produces a feasible solution over the 50 runs for candidate lists one and five. LE obtained the best result compared to the other heuristics due to the room related constraints (i.e. distance and splitting constraint). Having those two constraints reduces the effectiveness of SD and LD. This is perhaps not surprising as SD and LD are designed to specifically target spreading the examinations.

Table 4.1 Result using graph colouring heuristic

Graph Colouring Heuristic	Candidate list = 1				Candidate list = 5			
	Ave	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*
Largest degree (LD)	16.21	**1.52**	12.74	20.42	7.84	0.98	5.99	11.12
Largest weighted degree (LWD)	15.82	1.97	11.43	20.70	6.09	0.67	5.05	8.29
Largest enrolment (LE)	**15.51**	2.10	**10.98**	20.03	**6.06**	0.76	**4.74**	**7.98**
Saturation degree (SD)-LD	16.17	1.53	13.11	**19.39**	7.22	0.84	5.76	8.72
Saturation degree (SD)-LWD	16.29	1.54	13.97	20.41	7.00	1.02	5.49	9.78
Saturation degree (SD)-LE	16.09	1.80	12.66	20.74	6.96	**0.66**	5.28	8.49

Ave = average; *Stdev* = standard deviation; *Min* = minimum; *Max* = maximum

4.5 Contributions

The contributions of this work include collection of the necessary requirements (constraints) which has never before been properly documented at UMP. This data collection was carried out with the help and assistance of UMP employees. Studying the problem has led to two new constraints being identified; the travel distance for lecturers/invigilators and splitting exams across rooms. A further contribution of this work is the formulation of the UMP examination timetabling problem as a mathematical model. A simple yet effective approach of single or multiple room searching and selection is introduced through the pre-determined room grouping (which receive a positive respond from the EJOR reviewer). Finally, we have presented an algorithm, based on graph colouring heuristics, which we have shown can produce superior solutions compared to the software currently used. In addition,

the proposed algorithm adheres to all the hard constraints which the current methodology fails to do.

4.6 Conclusions

It is recognised that a gap exists between theory and practice in examination timetabling. Different institutions have different requirements and it is difficult to produce a common solution methodology. In this work we have introduced a new examination dataset with additional constraints (compared to the benchmark datasets). In particular, we have investigated the scheduling of exams in a capacitated environment with the aim of minimising the spreading, distance and splitting cost. A constructive heuristic has been used to generate solutions that produce better solutions when compared to the proprietary software that is used by UMP.

In the next chapter, we are going to schedule the invigilators to room/exam using the solution generated in this chapter as an input to that model.

Chapter 5

Universiti Malaysia Pahang Examination Timetabling Problem: Scheduling Invigilators

The work presented in this chapter is under review (after resubmission having addressed the reviewers comments) for the Journal of Operational Research Society (JORS). The problem involves assigning invigilators to examination rooms. This problem has not received the same level of research attention as other related problems, for example examination scheduling, but it is just as important to educational institutions. In modelling, and solving, this problem we assume that there is already an examination timetable in place (this was the subject of our previous work, see chapter 4) and the task is to assign invigilators to that timetable. The contributions of this chapter are to formally define the *invigilator-scheduling* problem and to present a constructive algorithm that is able to produce good quality solutions that are superior to the solutions produced when using the university's current software. The model we present, we believe, accurately reflects the real world problem capturing various aspects of the problem which have not been presented before in the scientific literature. Moreover, the proposed approach adheres to all hard constraints which the university's current system fails to do.

In section 5.1, we look at the examination timetabling problem particularly the *exam-timeslot-room* assignment. We present related work on invigilator scheduling in

section 5.2. A formal model of the problem is presented in section 5.3. In section 5.4, we describe the experimental setup for our proposed strategy. Section 5.5 gives a description of the dataset used in our experiments. In section 5.6, a comparison between the solutions achieved with the current method employed by UMP (which is produced using proprietary software), and our method, is presented in order to evaluate the effectiveness of the proposed methodology. We discuss the additional invigilator constraints and the results in sections 5.7 and section 5.8 respectively. In sections 5.9 and 5.10, we summarise our contribution and present our conclusions respectively.

5.1 Introduction

Many papers discussing the examination timetable problem can be found in the literature (i.e. PATAT conference paper). However, besides the problem of scheduling exams to timeslots and/or rooms, the educational examination timetabling problem does not end there. The problem also involves assigning invigilators to the exam/room. This is normally done after the institution has generated the *exam-timeslot-room* timetable (Burke et al., 1996). Most of the research found in the literature involves assigning exams to timeslots and/or rooms. Only a few papers have investigated invigilator scheduling (Burke et al., 1996; Ong, Liew and Sim, 2009, Cowling, Kendall and Hussin, 2002; Reis and Oliveira, 1999). One reason for invigilation scheduling receiving less attention from the research community is due to the fact that no datasets are available. In our view, there are three ways an instituition could implement invigilator assignment; by hiring outside staff, using their own staff or by using a mixture of in-house staff and outside staff. This point is further discussed in chapter 2.

This chapter investigates the invigilator scheduling problem taken from Universiti Malaysia Pahang (UMP). This invigilator dataset contains numerous constraints, which we believe have never been discussed or modelled before.

5.2 Invigilator scheduling

An exam timetable is often generated by first assigning exams to timeslots (*exam-timeslot* assignment). A further process then assigns rooms and/or invigilators after the exam timetable has been approved/accepted (Burke et al., 1996). It is evident in the literature that most published work only deals with *exam-timeslot* assignment. Only a few papers have addressed the *exam-room* assignment (Dammak, Elloumi and Kamoun, 2006) and very little work can be found on invigilator scheduling. A lecturer preference survey by Cowling, Kendall and Hussin (2002) reveals that:

- Invigilators prefer 2-3 invigilation duties with a one or two day gap between each duty.
- Lecturers with other responsibilities (e.g. administrative or research work) should be given a reduced number of invigilation duties.
- An adequate gap is given between invigilation duties and the lecturers' own papers. This is to allow the lecturers enough time to do their marking and submit their grades within the required time.
- A fair distribution of chief invigilator duties.

Ong, Liew and Sim (2009) developed an invigilation scheduling system concentrating on optimising lecturer preferences (i.e. invigilation dates, time and constraints) for UiTM Sarawak (Samarahan Campus). The invigilation scheduling only involves lecturers (see section 2.7.1; I2). Previously the schedule was prepared manually by the institution's invigilation scheduling committee. They randomly assigned invigilation duties and, later, there was a lot of swapping amongst the lecturers. This resulted in confusion, misunderstanding and complaints of uneven invigilation duty distribution. This motivated them to develop an invigilation system with the aim of optimising lecturer preferences. The system enables lecturers to view the examination timetable, choose their preferred invigilation timeslots, specify the examination date and the time of their own subjects; and view their individual schedule and the final exam/invigilation timetable. Reis and Oliveira (1999) experimented with an examination timetabling problem from the University Fernando Pessoa, Porto using constraint logic programming. They solve the problem by scheduling each exam into

an available timeslot. For each exam, one or several exam rooms are allocated and for each room, a set of invigilators is defined. The proposed approach included the following investigations:

- Scheduling exams into timeslots and, once completed, scheduling the rooms. Finally, they deal with invigilator scheduling.
- Schedule exams into rooms, then schedule the timeslot and then the invigilators.
- Schedule exams into timeslots, then schedule rooms and invigilators simultaneously.
- The exams, timeslots, rooms and invigilators are scheduled simultaneously.

A survey carried out by Awang et al. (2006) on the UMP examination timetable asked about invigilator satisfaction with their invigilation timetable. It revealed that most of the invigilators are not satisfied with the gap between invigilation duties and the number of invigilations. They suggested that each invigilation duty should have at least a 2 or 3 day gap. However, they prefer fewer invigilation duties, considering that they also need to be available/on-standby during their own exam paper. They requested an even spread of invigilation duties among the staff. As the timetable officer is open to any suggestions for improving the current timetable, we are motivated to include the suggestion above as an additional constraint in addition to the original constraints. These additional constraints are discussed further in section 5.7.

In this work, we solve the UMP examination timetable in two phases: firstly, we schedule the exams into timeslot and rooms simultaneously (Kahar and Kendall, 2010a), and presented in chapter 4. We then use the solution from the first phase as input to the invigilator scheduling phase. The scheduling of exams into timeslots, rooms and lastly the invigilators has been reported as the best sequence in order to produce a good quality solution (Reis and Oliveira, 1999). Our proposed approach to this second phase is presented in section 5.4, but first we describe the problem informally, and then present a formal definition.

5.3 Problem formulation

Indices

i, j $1...N$ where N is the number of examinations

l $1...L$ where L is the number of staff

r $1...R$ where R is the number of rooms

t $1...T$ where T is the number of timeslots

Parameters

S_l The status of staff l. 1 denotes a lecturer, 0 otherwise.

l_r The number of invigilators (including chief invigilator) required in each room r

a_{il} The exam-staff matrix where each element (denoted by $a_{il}, i \in \{1...N\}$ and $l \in \{1...L\}$) denoted as 1 corresponding as the staff teaches the course (or exam paper) in that semester, 0 otherwise.

Examination timetabling parameters

Note: These variables are set by the examination scheduling phase (see Kahar and Kendall 2010a)

x_{it} 1 if examination i is assigned to timeslot t, 0 otherwise

y_{ir} 1 if examination i is assigned to room r, 0 otherwise

z_{rt} 1 if room r is assigned to timeslot t, 0 otherwise

Decision variables

v_{lrt} 1 if staff l is assigned to invigilate in room r in timeslot t as an invigilator, 0 otherwise

w_{lrt} 1 if staff l is assigned to invigilate in room r in timeslot t as the chief invigilator, 0 otherwise

The objective function is as follows:

$$\text{Minimise, } F(x) = F_1 + F_2 \qquad \text{(Eq. 1)}$$

The first component of the objective function, F_1, is that the chief invigilator duties should be evenly spread among the lecturers in the staff list L ($S_l = 1$).

$$F_1 = \sum_{l=1}^{L} Chief\ duty(w_{lrt}) \qquad \text{(Eq.2)}$$

Where

$$Chief\ duty(w_{lir}) = \begin{cases} 0 & if \sum_{t=1}^{T}\sum_{r=1}^{R} w_{lrt} \le \lceil CId \rceil \\ 20 & otherwise \end{cases} \qquad \text{(Eq.3)}$$

The maximum number of chief invigilation duties assigned to every lecturer ($S_l = 1$) can be calculated based on the number of lecturers in the staff list L and the number of rooms selected in the *exam-timeslot-room timetable* solution. The number of chief invigilation duties is calculated by taking the ceiling value of *CId*. The calculation is as follows:

$$\lceil CId \rceil = \frac{\sum_{t=1}^{T}\sum_{r=1}^{R} z_{rt}}{\sum_{l=1}^{L} S_l} \qquad \text{(Eq.4)}$$

The second component of the objective function F_2 is concerned with the even spread of both invigilator and chief invigilator duties.

$$F_2 = \sum_{l=1}^{L} staff\ duty(v_{lir}, w_{lir}) \qquad \text{(Eq.5)}$$

Where

$$staff\ duty(v_{lir}, w_{lir}) = \begin{cases} 0 & if \sum_{t=1}^{T}\sum_{r=1}^{R} (v_{lrt} + w_{lrt}) \le \lceil Id \rceil \\ 20 & otherwise \end{cases} \qquad \text{(Eq.6)}$$

The maximum number of invigilation duties for all staff L can be calculated based on the number of invigilators (l_r) required in each room (from the *exam-timeslot-room timetable* solution) and the number of staff L. The required number of invigilation

duties for each member of staff is calculated by taking the ceiling value of *Id*. The calculation is as follows:

$$\lceil Id \rceil = \frac{\sum_{t=1}^{T}\sum_{r=1}^{R} z_{rt} l_r}{L} \qquad \text{(Eq.7)}$$

The objective function (Eq.1) is subject to the following constraints:

a) Invigilators cannot invigilate their own exam paper (H_I1).

$$\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{r=1}^{R}(a_{il}\, x_{it}\, y_{ir})(v_{lrt} + w_{lrt}) = 0 \qquad \text{For all } l \in \{1,...,L\} \qquad \text{(Eq.8)}$$

b) The chief invigilators must be a lecturer, $S_l = 1$ (H_I2).

$$w_{lrt} \le S_l \qquad \text{For all } l \in \{1,...,L\},\ t \in \{1,...,T\} \text{and } r \in \{1,...,R\} \qquad \text{(Eq.9)}$$

c) Staff are not assigned to more than one invigilation duty at a time (H_I3).

$$\sum_{r=1}^{R}(v_{lrt} + w_{lrt}) \le 1 \quad \text{For all } l \in \{1,...,L\} \text{ and } t \in \{1,...,T\} \qquad \text{(Eq.10)}$$

d) All staff are required to invigilate a maximum of three examinations within the exam period (H_I4).

$$\sum_{t=1}^{T}\sum_{r=1}^{R}(v_{lrt} + w_{lrt}) \le 3 \quad \text{For all } l \in \{1,...,L\} \qquad \text{(Eq.11)}$$

e) The total number of invigilators (including one as chief invigilator) assigned to each room r in timeslot t has to equal the number of invigilators required for each room l_r (H_I5).

$$\sum_{l=1}^{L}(v_{lrt} + 2w_{lrt}) = z_{rt}(l_r + 1) \quad \text{For all } r \in \{1,...,R\} \text{ and } t \in \{1,...,T\} \qquad \text{(Eq.12)}$$

Algorithm parameters:

- $l = 1...L$ where L is the number of staff available for the invigilation duties
- $r = 1...roomSelected$ where *roomSelected* is a list of selected rooms in the *exam-timeslot-room* assignment solution
- $m = 1...\ l_r$ *where* l_r is the number of invigilators required in room r
- $c = 1...C$ where C is the number of candidates list
- S_l status of staff (i.e. lecturer or other) l. 1 denoted as a lecturer, 0 otherwise.
- D_l holds the total invigilation duty for staff l.
- *totalCostValue*[c] store the cost value for assigning invigilator l to timeslot and room in candidates list c.
- $v_{lrt} = 1$ if staff l is assigned to invigilate in room r in timeslot t as an invigilator, 0 otherwise
- $w_{lrt} = 1$ if staff l is assigned to invigilate in room r in timeslot t as the chief invigilator, 0 otherwise

Step 1: Set-up
  Sort Staff L in ascending order based on D_l or randomly
  Calculate the ceiling value *ceilingCld (eq.4)* and *ceilingId (eq.7)*
Step 2: Assign chief invigilators to room
  Set r ← 1
  Until r = *roomSelected* repeat:
  (2.1) **Set** l ← 1
  (2.2) **Set** c ← 1
  (2.3) **Until** $c = C$, **repeat**:
    (2.3.1) **If** $l \leq L$ and l is a academic staff ($S_l = 1$), **then** calculate the cost value F and store in totalCostValue[c], simultaneously **s.t.** l does not teach the exam (H1), no other invigilation duty within the same timeslot(H3), does not exceed the maximum invigilation duty (H4) and Invigilator on duty during their exam must be on the same building (H6) – optional
    (2.3.2) Increase l
    (2.3.3) **If** $l > L$, set l ← 1, totalCostValue[c] ← +∞
  (2.4) Select the minimum total cost value from C, set w_{lrt}←l and update D_l if totalCostValue[c]≠ +∞ for every c
  (2.5) Increase r
  (2.6) **Sort** Staff L in ascending order based on D_l or randomly
Step 3: Assign invigilators to room
  Set r ← 1
  Until r = *roomSelected* repeat:
  (3.1) **Set** l ← 1
  (3.2) **Set** m ← 1
  (3.3) **Until** m = l_r-1 **repeat:**
    (3.3.1) **Set** c ← 1
    (3.3.2) **Until** $c = C$, **repeat**:
      (3.3.2.1) **If** $l \leq L$, **then** calculate the cost value F and store in totalCostValue[c], simultaneously increase c **s.t.** l does not teach the exam (H_l1), no other invigilation duty within the same timeslot(H_l3), does not exceed the maximum invigilation duty (H_l4) and Invigilator on duty during their exam must be on the same building (H_l6) – optional
      (3.3.2.2) Increase l
      (3.3.2.3) **If** $l > L$, set l ← 1, totalCostValue[c] ← +∞
    (3.3.3) Select the minimum total cost value from C, set v_{lrt}←l *and update* D_l if totalCostValue[c]≠ +∞ for every c
    (3.3.4) Increase m
  (3.4) Increase r
  (3.5) **Sort** Staff L in ascending order based on D_l or randomly
Step 4: Verification and Cost value
  Verify the solution and **Calculate** Cost Value (Eq.1)

Figure 5.1 Pseudocode for the invigilator scheduling

5.4 Experimental setup

In this section, we present our proposed invigilator scheduling algorithm in order to solve the UMP problem. As described previously, invigilator scheduling is a post-process from the *exam-timeslot-room timetable* process (Kahar and Kendall, 2010a). Therefore, the information (e.g. rooms, exams, timeslot etc.) from the *exam-timeslot-room* assignment phase is already known and, hence the results that produce the minimum cost value are retained from this first phase. Even if several runs were made in the first phase, the run that produced the minimum cost value is saved. Referring to section 5.3, the chief invigilator assignment is the most critical part as it involves the most constraints; must be a lecturer, cannot invigilate their own paper, etc. Invigilator assignment is less complicated as the member of staff can be a lecturer, or otherwise. Hence, we have designed an algorithm that firstly concentrates on assigning the chief invigilators to all the rooms, followed by other invigilator assignments.

The algorithm (see figure 5.1) starts (line 2) by sorting staff L in ascending order based on D_l or randomly. Next in line 3, we calculate the ceiling invigilation value for chief, *ceilingCId* (Eq.4) and invigilator duties, *ceilingId* (Eq.7) (see line 3). Then, we assign a chief invigilator into room in the *roomSelected* list (step 2, line 4). The first staff in L is selected. The number of chief invigilator we consider is referred to as candidates list (which we use during the random ordering strategies) and we show the effect of different candidates list sizes in the result section. If l is a lecturer ($S_l = 1$) and satisfies the following: l does not teach the exam (H_l1), has no other invigilation duty within the same timeslot (H_l3) and does not exceed the maximum number of chief invigilation duties (H_l4), we then calculate the penalty value on assigning the selected invigilator to r and store the information in *totalCostValue*[c] (lines 10–14). We also consider the invigilator should be in the same timeslot and building as their own exam if on duty during their exam constraint (H_l6) in this step during the additional constraints experiments. Next, increase c to search of other l for the candidates list. Then, increase l, however if l is greater than L, we set l =1 and assign *totalCostValue*[c] = +∞ (which means that there are no available invigilator in *totalCostValue*[c]) (lines 15). The search continues by selecting the minimum total cost value in C (i.e. *totalCostValue*[C]) and set the corresponding l into the selected timeslot and room, w_{lrt}

= 1 and subsequently increase D_l (lines 16). Finally, we increase r (line 17) and sort staff L in ascending order based on D_l (this would let the search to always select the minimum number of invigilation duties of staff L) or randomly (line 18).

Next, we assign the invigilators (step 3, line 19). The same process is carried out as for assigning chief invigilators except now, the search will continue for a l_r-1 of duration for each *roomselected* (line 24). l_r is the number of invigilators required in *roomselected.* For example, if l_r = 4, then the search will iterate 3 times (which is equivalent to three invigilators and one chief invigilator). Lastly, the algorithm verifies whether the solution complies with all the hard constraints and calculates the cost of the solution (line 38).

5.5 UMP invigilator dataset

Experiments were carried out with two different datasets from semester1-200708 and semester1-200809. The data is obtained from the solution generated by the UMP proprietary software. We noticed that there is a difference in the information (i.e. staff status, number of lecturers etc) provided by the Academic Office compared to the actual solution that they provided us with. Therefore, we decided to use the data from the schedule that was actually used as this more accurately represents what was done in practice. A description of the datasets is given below.

Table 5.1 Summary of UMP investigated datasets

Categories	Semester1-200708	Semester1-200809
Exams	157	165
Students	3,550	4,284
Enrolments	12,731	15,416
Conflict density	0.05 (5%)	0.05 (5%)
Timeslot per day	2	2
Rooms	24	28
Invigilator	152[a] 75[b]	207[a] 125[b]

[a] *number of lecturers,* [b] *number of non-lecturers*

Semester1-200708; the number of staff available for invigilation duties is 227. Of those, 152 are lecturers and 75 are non-lecturers. Each room must be allocated two invigilators (including the chief invigilator). 169 lecturers are involved in teaching the 157 exams. The 169 lecturers are not necessarily included in the staff list, *L*. In semester1-2008/09; the number of staff available for invigilation duty is 332. Of those, 207 are lecturers and 125 are non-lecturers. The total number of invigilators required by each room varies from a minimum of two to a maximum of four (including the chief invigilator). 194 lecturers are involved in teaching the 165 exams. The 194 lecturers are not necessarily included in the staff list, *L* because of other commitment during the exam week (e.g. administration task etc).

5.6 Results

In this section, we present the results of the invigilator timetable generated by the UMP proprietary software by inputting their solution into the model described in section 5.3. A comparison of the result obtained by the UMP proprietary software with our proposed algorithm (section 5.4) is also discussed. The results are summarised in table 5.2.

5.6.1 Semester1-200708

Analysing the solution produced by the UMP proprietary software in the *exam-timeslot-room* assignment phase, a total of 269 rooms were used. Therefore, using these 269 rooms the invigilator scheduling problem exhibits the following characteristics (see table 5.2, column A).

Hard Constraints: From the constraints in chapter 3, section 3.3.2 (page 60), the invigilator timetable produced by UMP only complies with two out of the five hard constraints violating the following:

i) Constraint, H_I1: Staff are assigned to invigilate their own exam paper. Supposedly, they need to be available during the exam of their own paper to answer any queries.

ii) Constraint, H_I4: Staff are assigned to more than three exams which exceeds the maximum number of invigilation duties within the exam period.

iii) Constraint, H_I5: one room was not assigned the required number of invigilators.

Soft Constraints: The objective of the invigilator scheduling solution is measured based on two objectives. The cost value for F_1 (eq.2) is 220 and F_2 (eq.5) is 20 with a total cost value of 240.

5.6.2 Semester1-200809

Based on the result produced by the UMP proprietary software, 290 rooms have been used. The invigilator scheduling solution for semester1-2008/09 exhibits the following characteristics (see table 5.2, column A).

Hard Constraints: The invigilator scheduling produced by UMP violates all five of the hard constraints listed in section 3.3.2.
Soft Constraints: The cost value of the invigilator timetable solution for F_1 (eq.2) is 20 and F_2 (eq.5) is 120 with a total cost value of 140.

5.6.3 Proposed solution approach

In scheduling invigilators, our experiments use the *exam-timeslot-room* solution produced by the UMP proprietary software for semester1-200708 and semester1-200809 (see table 5.2, column B). We also use a solution from our own approach based on a graph colouring heuristic approach (Kahar and Kendall, 2010a) (see table 5.2, column C). The experiments were run on a Pentium core2 processor. The average running time was about ≈23 seconds. However, the running time depends on the

number of rooms being selected in the *exam-timeslot-room* assignment phase. Obviously, a higher number of rooms would slightly increase the running time, but this is not of particular significance, given the nature of the problem being addressed.

Using least invigilation duties ordering strategies on the UMP solution from semester1-200708 (269 rooms) and semester1-200809 (290 rooms), our proposed approach shows that we are able to produce a solution that satisfies all the constraints (both hard and soft) with a zero cost value (see table 5.2, column B). Next, using the result from our graph colouring heuristic approach (Kahar and Kendall, 2010), our invigilator scheduling approach is also able to produce a feasible result with no cost value for both of the datasets (see table 5.2, column C).

Based on this result, it is clear that our proposed invigilator scheduling strategy produces a superior solution compared to the solution produced by the UMP proprietary software. We believe the success of the approach is because of the two-phase method that schedules the chief invigilator followed by the other invigilators. In addition, the ordering of least invigilation duty aids in efficiently selecting suitable invigilators while optimising the spread of invigilation duties (i.e. soft constraints, S_I1 and S_I2). In discussion with the UMP Academic Office, their poor solution is perhaps due to staff swapping their invigilator duties among themselves after the schedule is published. A common reason being that the invigilator is unsatisfied with their timetable (i.e. invigilation duties close to one another, unable to invigilate one (or more) of their own exams is scheduled on the same day etc.) and due to other commitments (e.g. meetings, administrative work etc.). The Academic Office will update the changes requested and these changes contribute to a poor solution. Currently, the system neglects the effect of moving or swapping (on request) the invigilation duties, which we will consider in our future work.

We notice that the invigilator scheduling solution depends on the number of rooms being selected in the *exam-timeslot-room* assignment phase. Recall that the total rooms selected from the proprietary software in semester1-200708 and semester1-200809 is 269 and 290 respectively. In our constructive phase (Kahar and Kendall, 2010a), the average percentage of rooms selected for semester1-200708 is 16% (i.e. 244) less and

for semester1-200809 it is 10% (i.e. 274) less compared to the UMP proprietary software. Obviously, having a lesser number of rooms selected (in the *exam-timeslot-room* assignment phase) would automatically minimise the invigilation duties for the staff.

Table 5.2 Invigilator scheduling results using constraint as describe in section 5.3

Constraints	(A) Proprietary software		(B) Our approach using exam timetable from UMP				(C) Our approach using exam timetable from Kahar and Kendall, 2010a			
	Sem1-200708 (269 rooms)	Sem1-200809 (290 rooms)	Sem1-200708 (269 rooms)		Sem1-200809 (290 rooms)		Sem1-200708 (244 rooms)		Sem1-200809 (274 rooms)	
			c1 ≈23s	*c5* ≈28s	*c1* ≈52s	*c5* ≈62s	*c1* ≈22s	*c5* ≈26s	*c1* ≈53s	*c5* ≈60s
H1) Invigilators or chief invigilators cannot invigilate their own exam paper.	*Not (1)*	*Not (2)*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
H2) Only allow staff with lecturer status to act as a chief invigilator.	*Yes*	*Not (1)*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
H3) Staffs are not assigned to more than one invigilation duty in one timeslot.	*Yes*	*Not (2)*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
H4) Staff can only invigilate a maximum of three examinations within the exam period.	*Not (1)*	*Not (6)*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
H5) Each room should be assigned the required number of invigilators (including a chief invigilator).	*Not (1)*	*Not (2)*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Cost value functions ($F = F_1 + F_2$)	240	140	0	0	0	0	0	0	0	0

c1 = candidates list of one; c5 = candidates list of five; Not (x) = Not comply (number of violations);

Yes = Comply;

In summary, we have demonstrated that the proposed invigilator scheduling approach is able to produce a feasible solution that adheres to all constraints without any cost to the objective function (even with a higher number of rooms as in the solution from semester1-200708 and semester1-200809). However, the invigilator scheduling result is dependent on the number of rooms being selected from the *exam-timeslot-room* assignment phase.

5.7 Additional UMP invigilator scheduling constraints

We include additional constraints in addition to the UMP original invigilator constraints as described in section 5.3. This is motivated by a survey from Awang et al. (2006) on the UMP invigilator scheduling problem which reveal that most of the invigilators are not satisfied with their current invigilation duties. According to Awang *et al.*, invigilators suggested that each invigilation duty should have at least a 2 or 3 day gap between them and also suggested having fewer invigilation duties, considering that they also need to be available/on-standby during their own exam paper. Additionally, the invigilator requested an even spread of invigilation duties among the staff (as we have considered in the original constraints - see F_1 and F_2 in section 5.3). Moreover, according to the timetable officer they often receive request for changes from the invigilators. The common reasons being invigilation duties are consecutive, are to close together, staff need to be on standby as more than one of their exams are scheduled together etc. We hope to satisfy the invigilators requests and minimise the request for changes to the schedule. The additional hard constraints for the UMP *invigilator-scheduling* problem are as follows:

H_16) Invigilators, with a lecturer status, on duty during their exam paper need to be scheduled in the same timeslot and building as their own exam paper. The formulation is as follows

$$\sum_{p=1}^{R}(v_{lrt}+w_{lrt})\cdot own(a_{il},x_{it},y_{ip})=(v_{lrt}+w_{lrt})\cdot a_{il}x_{it}m_i \quad \text{For all } l\in\{1,...,L\},$$

$$t\in\{1,...,T\}, r\in\{1,...,R\} \text{ and } i\in\{1,...,N\} \quad \text{(Eq.13)}$$

Where

$$own(a_{il},x_{it},y_{ip})=\begin{cases}1 & if\ (a_{il}x_{it}y_{ip})=1\ and\ (B_r=B_p)\\ 0 & otherwise\end{cases}$$

Where m_i is the number of rooms exam i has been split across and B_r is the building for room r. The additional soft constraints are as follows:

S_I3) Each invigilation duty should have at least 2 day gap, for every invigilator. A penalty is given if this is violated. The formulation is:

$$F_3 = \sum_{l=1}^{L}\sum_{r=1}^{R}\sum_{t=1}^{T}(v_{lrt} + w_{lrt}) \cdot \left(\sum_{t+s\,(where\ s=1)}^{s\le 5\ and\ t+s\le T} \sum_{p=1}^{R} gap(v_{lp(t+s)}, w_{lp(t+s)}) \right) \qquad \text{(Eq.14)}$$

Where

$$gap(v_{lr(t+s)}, w_{lr(t+s)}) = \begin{cases} \dfrac{32}{2^s} & if\ (v_{lr(t+s)} + w_{lr(t+s)}) = 1 \\ 0 & otherwise \end{cases}$$

Where s is a constant values of 1 to 5.

S_I4) There is a penalty associated with staff on duty during their exam paper. If the staffs are on duty during their exam, they need to be scheduled in the same timeslot and building as their exam; see hard constraint, H_I6.

$$F_4 = \sum_{i}^{N}\sum_{l}^{L}\sum_{r}^{R}\sum_{t}^{T}(v_{lrt} + w_{lrt}) \cdot duty(a_{il}, x_{it}) \qquad \text{(Eq.15)}$$

Where

$$duty(a_{il}, x_{it}) = \begin{cases} 3 & if\ (a_{il}x_{it}) = 1 \\ 0 & otherwise \end{cases}$$

A penalty value of three is chosen based on the feedback of the UMP timetable officer. According to the officer, the exam questions go through a series of checks (e.g. grammar checking and subject expert panel) in order to ascertain that it is error free. Therefore, the officer believes that this is not a major issue. However, it is preferable not to have the staff on duty during their exam paper. Hence, we believe that a value of three is adequate to represent the penalty.

5.8 Results for the additional invigilator constraints

We present the results of the invigilator timetable generated by the UMP proprietary software considering the additional constraint. In our proposed approach two different strategies were used which involve sorting the invigilators randomly and also sorting by the least number of invigilation duties. The results are summarised in tables 5.3 and 5.4.

5.8.1 Proprietary software results

In semester1-200708, considering the additional invigilator constraints, the solution exhibits the following characteristics. (see table 5.3, column A).

- Hard Constraint (H_i6): The UMP results violate the invigilators on duty during their own exam paper as they should be assigned in the same timeslot and building as their own exam paper.
- Soft Constraints: measuring the solution using the additional soft constraint results in a total of 369 with the cost value for F_3 (eq.14) = 120 and F_4 (eq.15) = 9 (the value of F_1 and F_2 remain the same).

In semester1-200809, the result shows that (see table 5.3, column-A):

- Hard Constraints (H_i6): The UMP results violate the constraint.
- Soft Constraints: The total cost value of the invigilator timetable solution is 713 with F_3 (eq.14) = 546 and F_4 (eq.15) = 27 (F_1 and F_2 remain the same).

5.8.2 Our approaches

We consider the additional invigilator constraints in scheduling the invigilators using the *exam-timeslot-room* solution produced by the UMP proprietary software for semester1-200708 and semester1-200809, and the solution from Kahar and Kendall

(2010a). The following discussion is based on the least invigilation duties ordering and random ordering approach.

5.8.2.1 Least invigilation duties ordering

The least invigilation duties selects the invigilator with the least duties. Using a candidate list of one for the UMP solutions from semester1-200708 (see table 5.3, column B), our proposed approach shows that we are able to produce a solution that satisfies all the hard constraints with a cost value of 978 (≈39sec). Increasing the candidate list to five, we manage to produce a slightly better solution with a cost value of 839 (≈83sec). For semester1-200809 dataset, using a candidate list of one the solution produced satisfies all the hard constraints with a cost value of 1634 (≈101sec) and with a candidate list of five, the cost value is 1419 (≈180sec). The results are summarised in table 5.3, column B. Comparing the above result with the proprietary software, although our result produces a high cost value (for both datasets), it does satisfy all of the hard constraints compare to the result from the proprietary software.

Next, using the result from our graph colouring heuristic (Kahar and Kendall, 2010) (see table 5.3, column C), for semester1-200708 with a candidate list of one, the solution produced satisfies all the hard constraints with a cost value of 860 (≈35sec). Increasing the candidate list to five, the cost value is 86 (≈74sec), 77% (369 compared with 86 ((369 – 86)/369 x 100%)) better than the UMP result. For the semester1-200809 dataset, using a candidate list of one the solution produced satisfies all the hard constraints with a cost value of 1092 (≈90sec) and with a candidate list of five, the cost value is 234 (≈165sec), that is 67% (713 compared with 234 ((713 – 234)/713 x 100%)) better than the UMP result. The results are shown in table 5.3 column C. Based on these results, using the approach presented in Kahar and Kendall (2010) to provide the examination timetable, the result we produce is superior to the UMP proprietary solution and also when using the UMP proprietary result, even when we include the additional constraints that are not presented in the proprietary software. We believe the reason for this is that having a lesser number of rooms used (see table 5.3), minimises the number of invigilation duties, thus allowing the duties to be spread out more fairly.

Table 5.3 Invigilator scheduling results with additional constraint using least duties ordering approach

Constraints	(A) Proprietary software		(B) Our approach using exam timetable from UMP				(C) Our approach using exam timetable from Kahar and Kendall, 2010			
	Sem1-200708 (269 rooms)	Sem1-200809 (290 rooms)	Sem1-200708 (269 rooms)		Sem1-200809 (290 rooms)		Sem1-200708 (244 rooms)		Sem1-200809 (274 rooms)	
			c1 ≈39s	*c5* ≈83s	*c1* ≈101s	*c5* ≈180s	*c1* ≈35s	*c5* ≈74s	*c1* ≈90s	*c5* ≈165s
H6) Invigilators on duty during their exam paper need to be schedule in the same timeslot and building as their own exam paper	*Not (1)*	*Not (6)*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Total cost value or violation of the soft constraint ($F = F_1$ to F_4)	369	713	978	839	1634	1419	860	86	1092	234

c1 = candidates list of one; c5 = candidates list of five; Not (x) = Not comply (number of violations); Yes = Comply

5.8.2.2 Random ordering

In using a candidate list of one on the UMP solutions from semester1-200708, our proposed approach shows that we are able to produce a solution that satisfies all the hard constraints with a minimum cost value of 2155 (see table 5.4, column A). Increasing the candidate list to five, the search produces far better minimum cost value of 201 that is 45% (369 compared with 201 ((369 – 201)/369 x 100%)) better when compared with the proprietary software and 76% (839 compared with 201 ((839 – 201)/839 x 100%)) better when compared to using least duties ordering. For semester1-200809 dataset (table 5.4, column A), using a candidate list of one the solution satisfies all the hard constraints with a minimum cost value of 2578. Using a candidate list of five, the minimum cost value is 190, 73% (713 compared with 190 ((713 – 190)/713 x 100%)) better when compared with the proprietary software and 87% (1419 compared with 190 ((1419 – 190)/1419 x 100%)) better when compared to using least duties ordering. Referring to the result above, with a candidate list of five, we are able to produce a good quality solution when compared to using a candidate list

of one, the UMP proprietary software and using a least duties ordering strategy. Additionally, candidates list of five is adequate as increasing the number of candidate list will enable better exploration of the search space but it would increase the computational time.

Next, using the result from our graph heuristic (Kahar and Kendall, 2010a), for semester1-200708 with a candidate list of one (see table 5.4, column B), the solution produced satisfies all the hard constraints with a minimum cost value of 1617. Increasing the candidate list to five, the solution has a minimum cost value of 67 that is 82% (369 compared with 67 ((369 – 67)/369 x 100%)) better when compared with the proprietary software and 22% (86 compared with 67 ((86 – 67)/86 x 100%)) better when compared to using least duties ordering. For semester1-200809 dataset (see table 5.4, column B), using a candidate list of one the solution produced satisfies all the hard constraints with a minimum cost value of 1918. Increasing to candidate list of five, the minimum cost value is 49, 92% (713 compared with 49 ((713 – 49)/713 x 100%)) better when compared with the proprietary software and 79% (234 compared with 49 ((234 – 49)/234 x 100%)) better when compared to using least duties ordering. Referring to the result above, our proposed approach is able to return a good quality solution (when using a candidate list of five). Overall, the least duties ordering approach produce a good quality solution, outperforming the proprietary software and random ordering (with a candidate list of one). However, the random ordering with a candidate list of five outperforms the least duties ordering approach. Based on our observation, this is because, in least duties ordering it will always select the result (invigilator) that returns a lower penalty value during the early stages of the search. However, towards the end of the search, the search becomes more difficult and the least duties ordering has a higher penalty cost (in order for feasible solution).

The proposed invigilator scheduling strategy is able to produce good quality solutions even with additional constraints (H_I6, S_I3 and S_I4). This demonstrates that we are able to produce a feasible solution and satisfy the additional invigilator requests (based on the comments of Awang *et al.* 2006) which we believe would benefit the timetable officer (rather than them need to respond to changes post schedule publication). In

summary, we have demonstrated that the proposed invigilator scheduling approach is able to produce a feasible solution that adheres to all constraints, including the additional constraints not previously captured.

Table 5.4 Invigilator scheduling results for additional constraint using random ordering approach

Constraints		(A) Our approach using exam timetable from UMP				(B) Our approach using exam timetable from Kahar and Kendall, 2010			
		Sem1-200708 (269 rooms)		Sem1-200809 (290 rooms)		Sem1-200708 (244 rooms)		Sem1-200809 (274 rooms)	
		c1 ≈39s	*c5* ≈83s	*c1* ≈101s	*c5* ≈180s	*c1* ≈35s	*c5* ≈74s	*c1* ≈90s	*c5* ≈165s
H6) Invigilators on duty during their exam paper need to be schedule in the same timeslot and building as their own exam paper		*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
	Stdev	135	47	143	27	130	31	157	20
Total cost value or violation of the soft constraint ($F = F_1$ to F_4)	Ave	2546	310	2886	246	1867	139	227	90
	Min	2155	201	2578	190	1617	67	1918	49
	Max	2784	406	3161	306	2180	200	2596	152

c1 = candidates list of one; c5 = candidates list of five; Not (x) = Not comply (number of violations);
Yes = Comply

5.9 Contributions

The contributions of the work include collection of the invigilator constraints which have never before been properly documented at UMP. We formulate the UMP invigilation scheduling problem as a formal model. The model presented here has never been modelled before in the literature. Additionally, we include additional constraints for invigilator scheduling. The additional constraints, we believe more accurately captures the UMP invigilation scheduling problem that is done at the moment. Finally, the proposed of a constructive technique that able to produces good quality solutions, satisfying all hard constraints (including the additional constraints) that the UMP proprietary system fails to do.

5.10 Conclusions

In this work, we have investigated invigilator scheduling for a real world examination timetabling problem, which aims to satisfy a number of constraints. The problem is complicated by the fact that the chief invigilator position can only be assigned to academic staff and staff are not allowed to invigilate their own papers. Furthermore, the invigilation duties assignment has to meet the required number of invigilators (including the chief invigilator) for each room avoiding clashes and complying with the maximum number of invigilation duties for each member of staff. A least ordering search was used to schedule the invigilators. The proposed approach is able to produce good quality solutions compared to the UMP proprietary software, satisfying all the constraints, both hard and soft, which the proprietary software fails to do. Additionally, we have included extra constraints, based on the comments in Awang *et al.*, (2006). Different ordering strategies (i.e. least duties and random ordering) have been used to schedule the invigilators. We have shown that a good quality solution can be produced even with these additional constraints. We believe that the solutions produced would satisfy all parties (i.e. officers and staff).

The next chapter, we are going to improve the result from constructive phase (in chapter 4) as it show that the invigilator assignment dependent on the number of room use. Hence it is best to optimise the *exam-timeslot-room assignment.*

Chapter 6

A Great Deluge Algorithm for a Real World Examination Timetabling Problem.

The work presented in this chapter is under review (after resubmission having addressed the reviewers comments) for the Journal of Operational Research Society (JORS). Many work found in the literature have been applied to simplified examination benchmark datasets. In this work we bridge the gap between research and practice by investigating a problem taken from the real world. This work introduces a modified and extended Great Deluge Algorithm (GDA) for the examination timetabling problem which uses a single, easy to understand parameter. We investigate different initial solutions, which are used as a starting point for the GDA, as well as altering the number of iterations. Additionally, we carry out statistical analysis to compare the results when using these different parameters. The proposed methodology is able to produce good quality solutions when compared to the solution currently produced by the host organisation, generated in our previous work and from the original GDA (Dueck, 1993).

Sections 6.1 give an introduction of the work carried out in this chapter. In section 6.2, we describe the GDA and our proposed modification. In section 6.3, we describe the experimental setup to allow reproducibility for other researchers. The result from the improvement phase is shown in section 6.4, followed by statistical analysis in section

6.5. Discussion on the results is presented in section 6.6. Lastly, in sections 6.7 and 6.8 we summarise the contribution and present our conclusions.

6.1 Introduction

In this work we present a modification of the great deluge algorithm (GDA) which allows the boundary, that acts as the acceptance level, to dynamically change during the search. The proposed algorithm will accept a new solution if the cost value is less than or equal to the boundary, which is lowered at each iteration according to a decay rate. The proposed GDA uses a simple parameter setting and allows the boundary to increase if there is no improvement after several iterations. Additionally, when the new solution is less than the desired value (estimation of the required cost value), the algorithm calculates a new boundary and a new desired value.

In order to investigate the proposed algorithm we use a real world capacitated examination problem taken from Universiti Malaysia Pahang (UMP). This dataset has several new constraints, in addition to those commonly found in the scientific literature. This work is an extension of our previous work described in Kahar and Kendall (2010a), in which we presented a constructive heuristic. We are now attempting to improve on the (initial) solution returned from the construction heuristic

6.2 Modified Great Deluge Algorithm (*modified*-GDA)

Suitable parameter settings are important in meta-heuristics and it is often difficult to determine the best values to guarantee a good quality solution (Petrovic and Burke, 2004). The introduction of a simple and easy to understood parameter (i.e. computational time and desired value) to determine the decay rate in Burke et al. (2004) made it straightforward for non-experts (e.g. university timetabling officers) to set the parameters, especially when compared to other meta-heuristic techniques (e.g. simulated annealing - cooling schedule, tabu search - tabu list size, genetic algorithm -

mutation or crossover probability rate etc.). Furthermore, they reported that their time-predefined GDA was able to produce good quality solutions.

The success of GDA and the simplicity in setting the parameters is the motivation for us to explore this method with the aim of bringing GDA to the university timetabling officer as they are the person responsible for producing the timetable at UMP. Our proposed GDA is shown in figure 6.1.

1. *Set the initial solution* s *from the constructive heuristic (Kahar and Kendall, 2010a)*
2. *Calculate initial cost function* $f(s)$
3. *Set the desired value* D
4. *Set the number of iterations* I
5. *Set Initial Boundary Level* $B = 0.03f(s) + f(s)$
6. *Set initial decay Rate* $\Delta B = (B - D)/I$
7. *Set* $s_{best} = s$
8. *While stopping criteria not met do*
9. *Apply neighbourhood heuristic on* s *to generate* s^*
10. *Calculate* $f(s^*)$
11. *If* $f(s^*) \leq f(s)$ *or* $f(s^*) \leq B$ *then*
12. *Accept* $s = s^*$
13. *If* $f(s^*) \leq f(s_{best})$ *then*
14. $s_{best} = s^*$
15. *Lower Boundary* $B = B - \Delta B$
16. *If no improvement in* W *iterations or* $B \leq f(s_{best})$ *or* $f(s) \leq D$ *then*
17. *Set* $s = s_{best}$
18. *If* $f(s) \leq D$ *then*
19. $D = f(s)*0.8$
20. *Set new decay rate* $\Delta B = (f(s) - D)/I_{remaining}$
21. *Set* $B = 0.03f(s) + f(s)$

Figure 6.1 Our proposed Great Deluge Algorithm

The algorithm starts by setting the desired value D, number of iterations I and the boundary level B (lines 1-5). The boundary level B is set slightly higher (3%) than the initial solution $f(s)$ obtained from a constructive heuristic (Kahar and Kendall, 2010a). The increment is to allow acceptance of worse result. We have tried several other percentages; a higher percentage leads to the search being unfocused, whilst a smaller percentage discourage exploration. Based on our observation, the 3% value is suitable for the investigated dataset. The decay rate is calculated as the difference between boundary level B and the desired solution D, divided by the number of iterations (line

6). While the stopping condition is not met, we apply the chosen neighbourhood heuristics. We calculate the new cost value $f(s^*)$ where $s^* \in N(s)$ selected at random (line 10). s^* is accepted if $f(s^*)$ is less than or equal to $f(s)$ or if $f(s^*)$ less than or equal to boundary B (lines 11-12). If $f(s^*)$ is less than or equal to $f(s_{best})$, set $s_{best} = s^*$ (line 13-14). Next, the boundary B is lowered based on the decay rate, ΔB (line 15). However, if there is no improvement for several iterations, W ($W = 5$ in this work) or boundary B is less than or equal to $f(s_{best})$ or $f(s)$ is less than or equal to desired value; then set $s = s_{best}$ (line 17). The new decay rate ΔB is calculated as the difference between $f(s)$ and desired value divided by the remaining number of iterations (line 20). However, if $f(s)$ is less than, or equal to, the desired value then a new desired value is calculated as 80% of $f(s)$ (line 18-19). This dynamically allows the search to continue with the search by having a new desired value. Based on our experiments a value above 0.8 unable to give a good result because of a steeper boundary (which discourage exploration). However values close to 0.8 able to give a relatively good result. Hence, the boundary is set 3% above $f(s)$ (line 21). Having this condition enables the algorithm to dynamically adjust the boundary, decay rate and desired value during the search.

We are going to compare the *modified*-GDA performance with the GDA propose by Dueck, (1993) (which will be refer to as *Dueck*-GDA in the following section), solution produced by UMP and from our previous work (Kahar and Kendall, 2010a).

6.3 Experimental setup

We implemented the proposed algorithm to the UMP semester1-200708 and semester1-200809 datasets. Descriptions of the dataset please refer to chapter 3. We run *Dueck*-GDA and our *modified*-GDA using several initial solutions selected randomly within the minimum to maximum values of the constructive solution presented in Kahar and Kendall (2010a). Note that, in Kahar and Kendall (2010a), the minimum and maximum values produced in semester1-200708 is 4.74 and 20.74 respectively, and in semester1-2008/09 it is 6.16 and 23.11 respectively. Hence, the (randomly) selected initial solutions in semester1-200708 is 16.68, 13.74, 10.30 and

7.82, and for semester 1-2008/09 they are 18.40, 15.25, 12.30 and 9.21. We ran both methods with 1500 and 3000 iterations. The following neighbourhood heuristics are used in our experiments. Note that, unless stated otherwise all the exam, timeslot and rooms are selected randomly.

Nh1) Move an exam to a different timeslot and room(s). This move is only possible when the destination room and timeslot is empty

Nh2) Move an exam to a different room(s) within the same timeslot.

Nh3) Move an exam to a different timeslot maintaining the currently assigned room(s)

Nh4) Choose an exam from a candidate list of 30, where exams are chosen based on their contribution to the objective function. An exam is chosen using roulette wheel selection and moved to a different timeslot and room(s).

Nh5) Same as Nh4 but move the exam to a different room(s) within the same timeslot

Nh6) Same as Nh4 but move the exam to a different timeslot maintaining the currently assigned room(s).

Nh7) Select two exams and swap the timeslot and room(s) between them.

Nh8) Select two timeslots and swap the timeslot between them

Nh9) Same as Nh4 but instead of moving the exam, we swap the selected exam with another exam.

Nh10) Select two timeslots and move all exams between the two timeslots. As an example if timeslot 2 and timeslot 6 were selected, move exams in timeslot 2 to timeslot 3; followed by moving exams in timeslot 3 to timeslot 4 and so on until exams in timeslot 6 are moved to timeslot 2.

In the next section we show the results when using each of these neighbourhoods.

6.4 Examination assignment: Results

In this section, we compare the examination timetable generated by the UMP proprietary software, our constructive heuristic (Kahar and Kendall, 2010a), and *Dueck*-GDA with our proposed GDA (*modified*-GDA). Each experiment was run 50 times on a Pentium core2 processor. The running time for 1500 iterations is around 480 seconds while 3000 iterations takes about 960 seconds. The result for semester1-200708 is shown in table 6.1 and semester1-200809 is shown in table 6.2.

6.4.1 Semester1-200708

6.4.1.1 *Modified*-GDA vs UMP proprietary software

The UMP proprietary software solution is 13.16 with a violation of one of the hard constraints (violating the no clash requirement H_E1, see chapter 3) (Kahar and Kendall, 2010a). Table 6.1 presents our results using the *modified*-GDA. Note that all of our results respect all the hard constraints. Using *modified*-GDA with 1500 iterations, we are able to produce a solution that is 66% (13.16 compared with 4.53 ((13.16 - 4.53)/13.16 x 100%)) better with Nh1 when using an initial solution with a cost of 7.82 compared to the solution produced by the proprietary software. The same calculation of percentage is used throughout the discussion. Increasing the number of iterations to 3000, the solution produced with Nh1, using an initial cost of 7.82, is 70% (13.16 compared with 4.01) better when compared to the proprietary software and 11% (4.53 compared with 4.01) better compared to using 1500 iterations. However, increasing the number of iterations, obviously, increases the computational time.

6.4.1.2 *Modified*-GDA vs constructive heuristic

In the constructive heuristic (Kahar and Kendall, 2010a) the best solution found was 10.98 and 4.74 using a candidate list of one and five respectively. Comparing *modified*-GDA with the constructive heuristic using a candidate list of one, in

Table 6.1 GDA result for semester1-200708

Neighbourhood moves	Initial Cost	*Modified*-GDA								*Dueck*-GDA							
		1500 iterations ≈ 8 min				3000 iterations ≈ 16 min				1500 iterations ≈ 8 min				3000 iterations ≈ 16 min			
		Ave	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*
Nh1) Move an exam to a different timeslot and room(s).	16.68	6.19	0.37	5.51	7.08	5.50	.30	4.99	6.03	13.72	0.39	12.80	14.40	12.92	0.31	12.22	13.53
	13.74	5.81	0.31	5.24	6.64	5.32	0.25	4.76	5.88	11.70	0.29	10.94	12.12	10.53	0.26	9.76	10.95
	10.30	5.16	0.26	4.65	5.67	4.59	0.23	4.10	5.04	8.79	0.11	8.49	9.02	7.47	0.06	7.29	7.57
	7.82	4.79	0.15	**4.53**	5.30	4.38	0.15	**4.01**	4.73	6.44	0.06	6.31	6.55	5.16	0.08	5.01	5.35
Nh2) Move an exam to a different room(s) within the same timeslot.	16.68	16.55	0.04	16.48	16.58	16.54	0.05	16.48	16.58	16.55	0.04	16.48	16.58	16.56	0.04	16.48	16.58
	13.74	13.22	0.00	13.21	13.22	13.22	0.00	13.21	13.22	13.22	0.00	13.21	13.22	13.22	0.00	13.21	13.22
	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30
	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82
Nh3) Move an exam to a different timeslot maintaining the current assigned room(s)	16.68	7.44	0.66	5.97	9.83	6.60	0.53	5.22	8.21	12.23	0.63	10.23	13.68	11.66	0.37	10.75	12.67
	13.74	6.90	0.31	6.12	7.61	6.34	0.27	5.84	6.89	11.13	0.36	10.03	11.81	10.13	0.25	9.58	10.55
	10.30	5.97	0.78	5.00	8.12	5.78	0.74	5.03	7.86	8.48	0.21	7.89	8.82	7.25	0.15	6.87	7.52
	7.82	5.71	0.33	4.90	6.39	5.49	0.32	4.79	6.06	6.41	0.23	6.16	7.82	5.15	0.19	4.80	5.86
Nh4) Move an exam selected among the highest penalty value to a different timeslot and room(s).	16.68	6.87	0.38	5.89	7.86	6.76	0.26	6.25	7.35	9.28	0.34	8.62	9.97	9.01	0.30	8.29	9.87
	13.74	6.76	0.38	5.88	7.58	6.64	0.45	5.77	7.88	9.48	0.34	8.79	10.22	9.15	0.33	8.50	9.71
	10.30	5.62	0.36	4.98	6.36	5.65	0.38	4.81	6.46	7.44	0.32	6.63	8.05	7.05	0.15	6.64	7.35
	7.82	5.22	0.19	4.90	5.73	5.22	0.21	4.57	5.84	6.22	0.13	5.90	6.47	5.62	0.24	5.03	6.50
Nh5) Same as in (Nh4) but move the exam to a different room(s) within the same timeslot	16.25	16.54	0.00	16.54	16.54	16.54	0.00	16.54	16.54	16.54	0.00	16.54	16.54	16.54	0.00	16.54	16.54
	13.74	13.50	0.01	13.49	13.51	13.50	0.01	13.49	13.51	13.50	0.01	13.49	13.51	13.50	0.01	13.49	13.51
	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30	10.30	0.00	10.30	10.30
	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82	7.82	0.00	7.82	7.82
Nh6) Same as in (Nh4) but move the exam to a different timeslot maintaining the current assigned room(s).	16.68	7.96	0.79	6.82	10.10	7.91	0.76	6.70	10.20	7.75	0.54	6.85	9.03	7.61	0.47	6.61	8.98
	13.74	8.21	0.59	7.13	9.22	7.97	0.67	7.05	9.78	8.45	0.40	7.78	9.18	8.05	0.39	7.26	8.99
	10.30	6.52	0.48	5.61	8.88	6.49	0.47	5.95	8.82	6.31	0.35	5.68	7.17	6.29	0.36	5.52	7.17
	7.82	6.19	0.30	5.40	6.92	6.07	0.38	4.79	6.83	5.89	0.48	**5.07**	7.31	5.96	0.45	5.16	7.00
Nh7) Select two exams randomly and swap the timeslot and room(s) between them	16.68	7.12	0.43	6.35	8.27	6.66	0.34	6.04	7.38	12.05	0.54	10.87	13.03	11.36	0.53	9.94	12.39
	13.74	7.37	0.43	6.53	8.85	6.97	0.43	6.20	8.08	10.85	0.40	9.83	11.52	10.02	0.27	9.22	10.65
	10.30	5.63	0.47	4.63	6.53	5.55	0.53	4.42	6.58	8.40	0.19	7.92	8.74	7.22	0.17	6.50	7.51
	7.82	5.40	0.31	4.66	6.01	5.09	0.30	4.66	5.79	6.29	0.12	5.91	6.47	5.17	0.15	**4.94**	5.54
Nh8) Select two timeslots and swap the timeslot between them.	16.68	8.45	0.38	7.68	9.48	8.36	0.43	7.40	9.59	9.50	0.45	8.25	10.82	8.93	0.35	8.19	9.67
	13.74	8.43	0.39	7.72	9.37	8.42	0.39	7.73	9.21	8.78	0.34	8.07	9.38	8.37	0.35	7.52	9.16
	10.30	7.20	0.33	6.64	7.82	7.17	0.28	6.66	7.83	7.25	0.34	6.67	8.03	7.13	0.31	6.57	7.81
	7.82	6.55	0.22	6.18	7.13	6.52	0.30	5.77	7.34	6.54	0.23	6.11	6.94	6.57	0.26	6.08	7.07
Nh9) Same as in (Nh4) but instead of moving exam, we swap the selected exam with another exam.	16.68	8.21	0.50	7.32	9.41	8.07	0.45	7.16	9.36	9.04	0.42	8.11	9.76	8.82	0.34	8.05	9.54
	13.74	8.13	0.68	6.91	9.77	7.76	0.47	6.90	8.86	8.47	0.48	7.49	9.54	8.10	0.25	7.52	8.61
	10.30	6.26	0.46	5.34	7.30	6.04	0.60	4.92	7.67	6.71	0.27	6.17	7.29	6.39	0.20	6.02	6.80
	7.82	6.02	0.28	5.44	6.72	6.04	0.35	5.35	6.78	5.79	0.20	5.38	6.41	5.43	0.30	4.91	6.14
Nh10) Select two timeslots and move all the timeslot between them.	16.68	8.77	0.46	7.79	9.88	8.60	0.44	7.82	9.61	9.77	0.48	8.58	10.65	9.11	0.41	8.19	9.88
	13.74	8.55	0.35	8.01	9.40	8.48	0.37	7.88	9.44	8.88	0.39	8.02	9.60	8.42	0.35	7.59	9.07
	10.30	7.48	0.32	6.79	8.34	7.31	0.38	6.57	8.44	7.32	0.31	6.65	8.10	7.13	0.33	6.57	7.99
	7.82	6.65	0.39	5.83	7.07	6.56	0.38	5.92	7.07	6.59	0.37	5.93	7.48	6.51	0.34	5.81	7.00

Ave = average; stdev = standard deviation; min = minimum; max = maximum

modified-GDA with 1500 iterations (table 6.1), we are able to produce a solution that is 59% (10.98 compared with 4.53) better with Nh1 using an initial solution of 7.82. Even with a poorer initial cost of 16.68, we are still able to improve the solution by 50% (10.98 compared with 5.51) with Nh1. Extending the search to 3000 iterations, initial cost of 7.82 and 16.68, Nh1 produced solutions with a 63% (10.98 compared with 4.01) and 55% (10.98 compared with 4.99) improvement when compared to the constructive heuristic solution.

In the constructive heuristic, with a candidate list of five, *modified*-GDA able to produce a better solution but with a small margin of improvement. Using an initial cost of 7.82 with 1500 and 3000 iterations, the GDA solution outperforms the constructive heuristic by 4% (4.74 compared with 4.53) and 15% (4.74 compared with 4.01) respectively. However, using a large initial cost 16.68, with 1500 and 3000 iterations, the constructive heuristic outperforms the *modified*-GDA by 14% (5.51 compared with 4.74) and 5% (4.99 compared with 4.74) respectively.

6.4.1.3 *Modified*-GDA vs *Dueck*-GDA

In the *Dueck*-GDA approach, with 1500 iterations it able to produce 5.07 cost value using Nh6 and with 3000 iteration produce 4.94 with Nh7. Comparing *modified*-GDA and Dueck-GDA with 1500 iterations (table 6.1), the *modified*-GDA able to produce a solution that is 11% (5.07 compared with 4.53) better than *Dueck*-GDA with Nh1 using an initial solution of 7.82. Even though with a poorer initial cost of 16.68, the *modified*-GDA were able to outperform *Dueck*-GDA by 20% (6.85 compared with 5.51) with Nh1. Extending the search to 3000 iterations, initial cost of 7.82 and 16.68, Nh1 produced solutions with a 19% (4.94 compared with 4.01) and 25% (6.61 compared with 4.99) improvement when compared to *Dueck*-GDA. The best values found by each of the method describe above is shown in figure 6.2.

Overall the proposed *modified*-GDA gives an improvement when compared to the UMP proprietary software, the constructive heuristic and *Dueck*-GDA. From these result it appears that using a better quality initial cost outperforms both the UMP

proprietary software and the constructive heuristic, but using a poorer quality initial solution, the *modified*-GDA does not guarantee to produce high quality solutions even for *Dueck*-GDA within our experimented number of iteration (when compared to the constructive heuristic with a candidate list of 5).

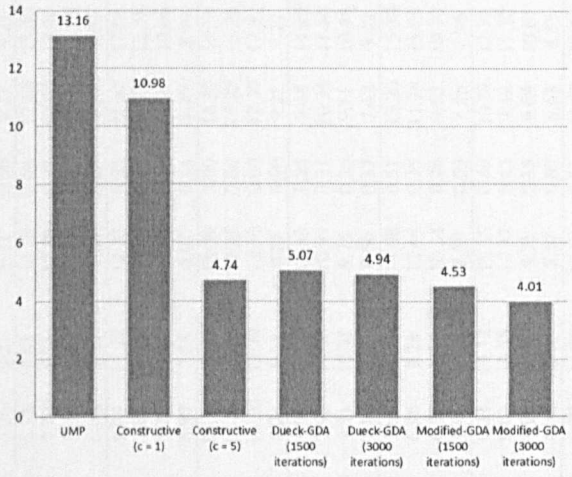

Figure 6.2 Best values of each method for semester1-200708

6.4.2 Semester1-200809

6.4.2.1 *Modified*-GDA vs UMP proprietary software

In semester1-200809 (refer table 6.2), the calculated UMP solution was 26.08 with a violation of all of the hard constraints (Kahar and Kendall, 2010a). The *modified*-GDA, with 1500 iterations, the solution produced is 77% (26.08 compared with 6.11) better compared to the proprietary software solution (and the solution adheres to all the hard constraints) using Nh1 with an initial cost of 9.21. Increasing the number of iterations to 3000, the solution produced with Nh1, using an initial solution of 9.21 is 78% (26.08 compared with 5.63) better than the proprietary software and 9% (6.11 compared with 5.63) better compared to using 1500 iterations.

Table 6.2 GDA result for semester1-200809

Neighbourhood moves	Initial Cost	*Modified*-GDA								*Dueck*-GDA							
		1500 iterations ≈ 8 min				3000 iterations ≈ 16 min				1500 iterations ≈ 8 min				3000 iterations ≈ 16 min			
		Ave	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*	*Ave*	*Stdev*	*Min*	*Max*
Nh1) Move an exam to a different timeslot and room(s).	18.40	8.42	0.33	7.61	9.37	7.55	0.28	6.96	8.26	15.67	0.40	14.76	16.48	14.78	0.24	14.30	15.27
	15.25	7.61	0.31	6.89	8.18	7.07	0.30	6.29	7.74	13.37	0.23	12.44	13.78	12.24	0.13	11.96	12.49
	12.30	6.89	0.24	6.33	7.61	6.38	0.20	6.03	6.89	9.50	0.06	9.36	9.61	9.48	0.08	9.22	9.61
	9.21	6.44	0.20	**6.11**	6.87	6.04	0.15	**5.63**	6.42	7.91	0.06	7.74	8.04	6.73	0.13	6.47	7.10
Nh2) Move an exam to a different room(s) within the same timeslot.	18.40	18.34	0.01	18.32	18.34	18.34	0.00	18.33	18.34	18.34	0.00	18.33	18.34	18.34	0.01	18.32	18.34
	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25
	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30
	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21
Nh3) Move an exam to a different timeslot maintaining the current assigned room(s)	18.40	8.00	0.36	7.12	8.69	7.26	0.30	6.78	8.03	14.33	0.66	12.68	15.50	13.68	0.40	12.86	14.36
	15.25	8.81	0.33	8.07	9.85	8.23	0.28	7.70	8.71	12.86	0.33	12.00	13.40	11.91	0.26	11.09	12.27
	12.30	8.25	0.35	7.66	8.99	8.02	0.33	7.32	8.80	10.64	0.22	10.05	10.95	9.43	0.09	9.19	9.65
	9.21	7.55	0.25	6.95	8.25	7.13	0.31	6.38	7.82	7.85	0.08	7.53	7.98	6.68	0.14	6.49	7.26
Nh4) Move an exam selected among the highest penalty value to a different timeslot and room(s).	18.40	10.15	0.52	9.11	11.41	9.91	0.52	9.09	11.22	12.33	0.35	11.42	13.01	12.07	0.34	11.21	12.81
	15.25	9.40	0.41	8.52	10.25	9.28	0.47	7.92	10.06	11.49	0.44	10.44	12.48	11.30	0.32	10.28	12.01
	12.30	8.46	0.31	7.66	9.18	8.38	0.37	7.72	9.55	10.18	0.24	9.66	10.66	9.56	0.16	9.30	9.97
	9.21	7.47	0.18	7.14	7.84	7.48	0.20	6.97	7.85	8.07	0.14	7.79	8.40	7.91	0.23	7.37	8.38
Nh5) Same as in (Nh4) but move the exam to a different room(s) within the same timeslot	18.40	18.39	0.00	18.39	18.39	18.39	0.00	18.39	18.39	18.39	0.00	18.39	18.39	18.39	0.00	18.39	18.39
	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25	15.25	0.00	15.25	15.25
	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30	12.30	0.00	12.30	12.30
	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21	9.21	0.00	9.21	9.21
Nh6) Same as in (Nh4) but move the exam to a different timeslot maintaining the current assigned room(s).	18.40	10.98	1.09	9.15	14.57	10.24	0.76	8.98	11.72	11.33	1.25	9.48	14.99	10.66	1.08	9.28	14.71
	15.25	10.98	0.24	10.30	11.61	9.28	0.47	7.92	10.06	10.99	0.31	10.34	11.54	10.88	0.25	10.12	11.37
	12.30	9.80	0.30	9.29	10.54	9.80	0.31	9.21	10.41	9.75	0.35	9.11	10.48	9.75	0.34	9.00	10.46
	9.21	8.96	0.20	8.15	9.14	9.00	0.22	7.86	9.21	8.50	0.37	7.62	9.02	8.50	0.38	7.71	9.09
Nh7) Select two exams randomly and swap the timeslot and room(s) between them	18.40	8.99	0.39	8.34	10.01	8.53	0.29	7.99	9.30	14.08	0.51	13.20	15.30	13.62	0.40	11.98	14.39
	15.25	8.21	0.39	7.57	9.27	7.79	0.40	6.96	9.28	12.44	0.35	11.74	13.03	11.68	0.21	11.05	12.05
	12.30	7.03	0.26	6.62	8.07	6.73	0.25	6.19	7.40	10.24	0.26	9.52	10.70	9.13	0.19	8.66	9.45
	9.21	6.71	0.27	6.21	7.35	6.37	0.24	5.86	6.85	7.79	0.09	7.56	7.96	6.57	0.09	**6.39**	6.85
Nh8) Select two timeslots and swap the timeslot between them.	18.40	10.79	0.45	9.39	11.90	10.23	0.45	9.31	11.12	11.35	0.49	10.31	12.73	10.68	0.43	9.93	11.79
	15.25	10.38	0.43	9.05	11.27	9.79	0.50	9.02	10.88	10.32	0.43	9.49	11.12	9.85	0.46	8.87	11.02
	12.30	9.66	0.34	8.96	10.31	9.69	0.34	8.98	10.36	9.71	0.35	9.03	10.61	9.60	0.31	8.97	10.70
	9.21	8.57	0.06	8.48	8.75	8.58	0.06	8.48	8.76	8.57	0.07	8.48	8.75	8.56	0.05	8.48	8.64
Nh9) Same as in (Nh4) but instead of moving exam, we swap the selected exam with another exam.	18.40	10.96	0.52	9.79	11.93	10.74	0.52	9.72	11.99	11.81	0.36	11.00	12.72	11.46	0.40	10.51	12.21
	15.25	9.88	0.38	8.97	10.64	9.85	0.37	9.09	10.53	10.63	0.36	9.96	11.45	10.22	0.35	9.48	10.93
	12.30	8.49	0.31	7.82	9.15	8.39	0.36	7.67	9.38	8.79	0.32	7.70	9.52	9.72	0.26	9.16	10.55
	9.21	7.52	0.24	7.02	8.02	7.45	0.23	6.78	7.87	7.55	0.17	**7.20**	7.92	7.39	0.23	7.01	7.89
Nh10) Select two timeslots and move all the timeslot between them.	18.40	10.59	0.45	9.51	11.67	10.40	0.47	9.52	11.84	11.42	0.45	10.29	12.33	10.74	0.45	9.92	11.93
	15.25	10.23	0.49	9.25	11.41	10.06	0.49	8.96	10.99	10.49	0.34	9.66	11.41	10.06	0.43	9.10	10.85
	12.30	9.73	0.32	9.26	11.15	9.67	0.20	9.38	10.21	9.70	0.23	9.16	10.50	9.59	0.27	8.94	10.33
	9.21	8.49	0.11	8.30	8.73	8.48	0.09	8.30	8.76	8.50	0.09	8.30	8.66	8.46	0.07	8.30	8.61

Ave = average; var = variance; stdev = standard deviation; min = minimum; max = maximum

6.4.2.2 *Modified*-GDA vs constructive heuristic

In the constructive heuristic (Kahar and Kendall, 2010a), the minimum solution produced is 13.89 and 6.61 using candidate lists of one and five respectively. In a comparison between *modified*-GDA and the constructive heuristic with a candidate list of one, the *modified*-GDA with 1500 iterations, produced a 56% (13.89 compared with 6.11) better solution with Nh1 using an initial cost of 9.21. Even with a poorer initial cost (18.40), the GDA solution is 46% (13.98 compared with 7.12) better using Nh3. Extending the search to 3000 iterations, when using an initial cost of 9.21 and 18.40, Nh1 produces 59% (13.89 compared with 5.63) and 51% (13.89 compared with 6.78), respectively, better solutions compared to the constructive heuristic.

Comparing the *modified*-GDA result with the constructive heuristic with a candidate list of five, *modified*-GDA with 1500 iterations outperforms the constructive heuristic by 8% (6.61 compared with 6.11). However, using a poorer initial cost (18.40), the constructive heuristic outperforms *modified*-GDA by 7% (7.12 compared with 6.61). In *modified*-GDA with 3000 iterations, it produces a 15% (6.61 compared with 5.63) better solution compared to the constructive heuristic. However, with the poorer initial cost (18.40), the constructive heuristic outperforms *modified*-GDA by just under 3% (6.78 compared with 6.61).

6.4.2.3 *Modified*-GDA vs *Dueck*-GDA

For *Dueck*-GDA, with 1500 iterations it able to produce 7.20 cost value using Nh9 and with 3000 iteration produce 6.39 with Nh7 (see table 6.2). Comparing *modified*-GDA and *Dueck*-GDA with 1500 iterations (table 6.2), the *modified*-GDA able to produce a solution that is 15% (7.20 compared with 6.11) better than *Dueck*-GDA with Nh1 using an initial solution of 7.82. With poorer initial cost of 16.68, the *modified*-GDA were able to outperform *Dueck*-GDA by 20% (9.48 compared with 7.12) with Nh3. Extending the search to 3000 iterations, initial cost of 7.82 and 16.68, *modified*-GDA with Nh1 produced solutions with a 19% (6.39 compared with 5.63) and 27% (9.28

compared with 6.78) improvement when compared to *Dueck*-GDA. The best value found by each of the methods described above is shown in figure 6.3.

Overall our proposed *modified*-GDA is able to generate superior solutions than the UMP proprietary software, the constructive heuristic (see Kahar and Kendall, 2010a) and *Dueck*-GDA. Based on the result from both datasets, it shows that using a good quality, initial solution will produce superior results, and possibly even better when using a larger number of iterations. This is possible because by using a good quality solution would allow the search to focus on the promising areas of the search space (Burke and Newall, 2002). In the next section, we will analyse the results.

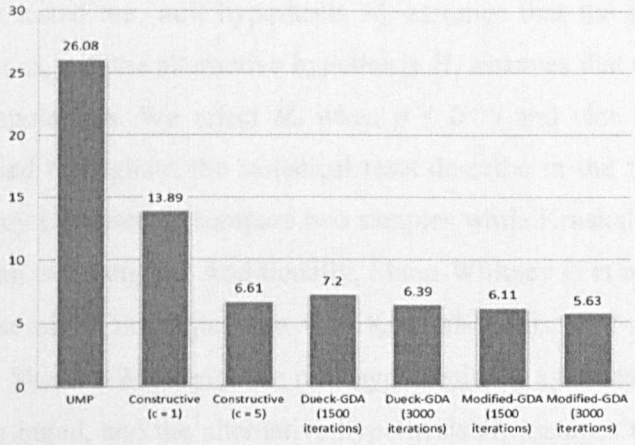

Figure 6.3 Best values of each method for semester1-200809

6.5 Statistical analysis

This section presents a statistical analysis of our results. The aim is to compare the *modified*-GDA and *Dueck*-GDA as well as the parameters used in the experiments to ascertain whether there are statistical differences. In addition we will determine suitable parameter values and neighbourhood heuristics. The comparisons include:

a) Compare different initial solutions: Is there any significant difference in using an initial solution with a higher cost than using a better quality initial solution?

b) Compare the number of iterations: Is there any significant difference in using a larger number of iterations?
c) Compare neighbourhood heuristics: Is there any significant difference in the result produce by using different neighbourhood heuristics?

Note that the analyses in (a) to (c) concentrate on the *modified*-GDA only.

We are conscious that some of these may seem intuitively obvious (e.g. increasing the number of iterations produces superior results) but it is still informative to do the analysis as it is often not carried out. A statistical test is carried out using Kruskal-Wallis and Mann-Whitney U to determine if there are significant differences. The hypotheses to be tested are, null hypothesis H_0 assumes that the samples are from identical populations, and the alternative hypothesis H_1 assumes that the sample comes from different populations. We reject H_0 when $p \leq 0.05$ and vice versa. The above hypothesis are used throughout the statistical tests describe in the following section. The Mann-Whitney U is used to compare two samples while Kruskal-Wallis is used to compare more than two samples. Additionally, Mann-Whitney U is used to investigate the rejection cause of H_0 in conjunction with Kruskal-Wallis. The normality test are carried out using Shapiro-Wilk with the null hypothesis H_0 assumes that the samples are normally distributed, and the alternative hypothesis H_1 assumes that the sample is non-normal. We reject the H_0 when $p \leq 0.05$ and vice versa.

We start the statistical test with a normality test using Shapiro-Wilk and continue with the relevant statistical test (as described above) based on the normality test result.

6.5.1 Semester1-200708

6.5.1.1 Significance difference: *Modified*-GDA and *Dueck*-GDA

We analyses the *modified*-GDA and *Dueck*-GDA result using Mann-Whitney U. Table 6.3 and table 6.4 show the *p*-value result for 1500 iterations and 3000 iterations respectively. For 1500 iterations, (see table 6.3), we notice that most of the result shows significant difference except for the Nh2 (all initial), Nh5 (all initial), Nh6

(16.68), Nh8 (10.30 and 7.82) and Nh10 (7.82). For 3000 iterations (see table 6.4), again most of the result show significant difference except for Nh2 (all initial), Nh5 (all initial), Nh6 (13.74, 7.82), Nh8 (13.74, 10.30, 7.82) and Nh10 (13.74, 7.82).

Based on both of the runs, generally the result that shows no significant difference involves neighbourhood heuristic that performs poorly with respect to quality of the obtained final solution (see table 6.1 and table 6.2).

Table 6.3 Semester1-200708 *p*-values comparison between *modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 1500 iterations

Neighbourhood heuristics	Initial cost			
	16.68	13.74	10.30	7.82
Nh1	.000	.000	.000	.000
Nh2	.694	.221	1.00	1.00
Nh3	.000	.000′	.000	.000
Nh4	.000	.000′	.000	.000
Nh5	1.00	.385	1.00	1.00
Nh6	.299	.037	.009	.000
Nh7	.000	.000	.000	.000
Nh8	.000	.000	.517	.900
Nh9	.000	.007	.000	.000
Nh10	.000	.000	.012	.251

Table 6.4 Semester1-200708 *p*-values comparison between *modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 3000 iterations

Neighbourhood heuristics	Initial cost			
	16.68	13.74	10.30	7.82
Nh1	.000	.000	.000	.000
Nh2	.019	.427	1.00	1.00
Nh3	.000	.000	.000	.000
Nh4	.000	.000	.000	.000
Nh5	1.00	.688	1.00	1.00
Nh6	.043	.115	.024	.095
Nh7	.000	.000	.000	.034
Nh8	.000	.634	.482	.296
Nh9	.000	.000	.000′	.000
Nh10	.000	.649	.013	.652

6.5.1.2 Comparing initial costs

We compare the initial cost based on the number of iterations for all neighbourhood heuristics. We use Kruskal-Wallis to compare between the initial costs (i.e. 16.68, 13.74, 10.30 and 7.82). At the 95% confidence interval, the statistical test shows that there is a difference (reject H_0) among the results produced between the initial costs for all of the neighbourhood heuristics (see table 6.5). Referring to table 6.5, the p-values are all less than 0.05 which leads us to reject H_0.

Table 6.5 Semester 1-200708 p-value comparison for the initial cost for each neighbourhood heuristic based on the number of iterations

Neighbourhood heuristics	*p-value*	
	1500 iterations	3000 iterations
Nh1	.000	.000
Nh2	.000	.000
Nh3	.000	.000
Nh4	.000	.000
Nh5	.000	.000
Nh6	.000	.000
Nh7	.000	.000
Nh8	.000	.000
Nh9	.000	.000
Nh10	.000	.000

In-depth analyses (see table 6.1) on the differences in pair (16.68 with 13.74, 10.30, 7.82; 13.74 with 10.30, 7.82 and so on) were investigated using Mann-Whitney U. Based on the analysis only a few of the initial cost shows no differences (accept H_0) which include:

- Nh3 between 10.30 and 7.82 for both iterations counts.
- Nh4 between 16.68 and 13.74 for both iterations counts.
- Nh6 between 16.68 and 13.74 with 3000 iterations.
- Nh8 between 16.68 and 13.74 for both iterations counts.
- Nh9 between 16.68 and 13.74 with 1500 iterations.
- Nh9 between 10.30 and 7.82 with 3000 iterations.
- Nh10 between 16.68 and 13.74 with 3000 iterations

Generally, the results that show no difference (accept H_0) involve using a solution with a large initial cost as well as neighbourhood that is underperformed. Hence, from this analysis we can conclude that it is more important to have a good neighbourhood while having a good solution helps to speed up the search process.

6.5.1.3 Comparing the number of iterations

We compare the number of iterations (1500 and 3000 iterations) based on the initial cost (i.e. 1500 vs 3000 with an initial cost of 16.68, 13.74, 10.30 and 7.82) using Mann-Whitney U. Table 6.6 shows the *p*-value of the comparison between the number of iterations executed. At the 95% confidence interval, the result is as follows (see table 6.6):

- Nh1 show significant difference (reject H_0) across all initial costs.
- Nh3 and Nh7 shows significant differences (reject H_0) for all initial costs except for 10.30 (accept H_0).
- Nh2, Nh4 and Nh8 show no significant differences (accept H_0) across all initial costs.
- Nh5 and Nh6 shows no significant differences (accept H_0) for all initial costs except during initial 13.74 (reject H_0).
- Nh9 show no significant differences (accept H_0) for all initial costs except for 13.74 and 10.30 (reject H_0)
- Nh10 show no significant differences (accept H_0) for all initial costs except during 10.30 (reject H_0).

Based on these tests, the result varies according to the neighbourhood heuristics. We notice that, an explorative neighbourhood heuristics (i.e. Nh1 and Nh7) show significance difference (reject H_0) between the two iterations compared to undiversified neighbourhood (i.e. Nh2, Nh5 etc). Therefore, (considering the solution in table 6.1) we conclude that it is best to use a large number of iterations. However, a search with a large number of iteration would only be worthwhile if it is being complemented with a good neighbourhood heuristic (to encourage exploration).

Table 6.6 Semester1-200708 *p*-value comparison between 1500 and 3000 iterations for each neighbourhood heuristic based on initial cost

Neighbourhood heuristics	Initial cost			
	16.68	13.74	10.30	7.82
Nh1	.000	.000	.000	.000
Nh2	.087	.340	1.00	1.00
Nh3	.000	.000	.051	.002
Nh4	.141	.127	.815	.702
Nh5	1.00	.038	1.00	1.00
Nh6	.860	.044	.524	.104
Nh7	.000	.000	.467	.000
Nh8	.236	.866	.692	.589
Nh9	.202	.006	.022	.495
Nh10	.061	.303	.005	.172

6.5.1.4 Comparing neighbourhood heuristics

We compare the entire neighbourhood heuristics based on the initial cost and number of iterations using Kruskal-Wallis (i.e. Nh1 vs Nh2 vs Nh3 vs ... Nh10 using initial cost 16.86 with 1500 iterations; etc). Table 6.7 show the *p*-values of the neighbourhood heuristics comparison. The result shows that there are significant differences (reject H_0) for the solutions produced using different neighbourhood heuristics.

Pair-wise comparison (analysis on the cause of H_0 rejection) using Mann-Whitney U on the neighbourhood heuristics show that there are significant differences (reject H_0) for the solution produced by most of the neighbourhood heuristics except for some. For example, Nh2 and Nh5 show no difference with an initial cost 7.82 and 10.30 for both iterations and initial cost 16.68 using 3000 iterations. Table 6.8 shows a summary of the non-significant differences (accept H_0) between the neighbourhood heuristics. Referring to table 6.8, we notice that, some of the neighbourhoods (i.e. Nh3 and Nh4, Nh4 and Nh7) show similarity although the inner working of the heuristics are different.

Table 6.7 Semester 1-200708 *p*-value comparison for the neighbourhood heuristics based on the initial cost and the number of iterations

Initial value	1500	3000
16.68	.000	.000
13.74	.000	.000
10.30	.000	.000
7.82	.000	.000

Finally, we can summarise that Nh1 produces the best result follow by Nh7 and Nh4. Next are Nh3, Nh9, Nh6, Nh8, Nh10, Nh2 and Nh5. In our observation, Nh1 is a robust neighbourhood heuristic. Nh2 and Nh5 are the worst neighbourhood heuristics as they are unable to give any improvement on the initial cost during the search (especially Nh5). Nh7 works best with a better quality initial cost, while Nh4 work best with a large initial cost. Further discussion on the neighbourhood heuristics is given in section 6.5.2.4.

Table 6.8 Semester1-2007/08 summary of the non-significant differences (accept H_0) when comparing the neighbourhood heuristics

	1500 iterations				3000 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
Nh1	-	-	-	-	-	-	-	-
Nh2	-	-	Nh5	Nh5	Nh5	-	Nh5	Nh5
Nh3	-	Nh4	Nh4	-	Nh4, Nh7	-	Nh4, Nh7	-
Nh4	-	-	Nh7	-	Nh7	-	Nh7	-
Nh5	-	-	-	-	-	-	-	-
Nh6	-	Nh8, Nh9	-	-	Nh9	Nh9	-	Nh9
Nh7	-	-	-	-	-	-	-	-
Nh8	-	Nh10	-	Nh10	-	Nh10	Nh10	Nh10
Nh9	-	-	-	-	-	-	-	-
Nh10	-	-	-	-	-	-	-	-

'-' = result show rejecting H_0

6.5.2 Semester1-200809

6.5.2.1 Significance difference: *modified*-GDA and *Dueck*-GDA

Same as in previous section (6.5.1.1), we used Mann-Whitney U to analyses the result. Table 6.9 and table 6.10 show the *p*-value result for 1500 iterations and 3000 iterations respectively. For 1500 iterations, (see table 6.9), we notice that most of the result shows significant difference except for the Nh2 (all initial), Nh5 (all initial), Nh6 (18.40, 15.25 and 12.30), Nh8 (15.25, 12.30 and 9.21), Nh9 (9.21) and Nh10 (12.30 and 9.21).

In 3000 iterations (see table 6.10), most of the result show significant difference except for Nh2 (all initial), Nh5 (all initial), Nh6 (12.30), Nh8 (15.25, 12.30 and 9.21), Nh9 (12.30 and 9.21) and Nh10 (15.25, 12.30 and 9.21).

Based on the result, semester1-200809 dataset show more non-significant difference compare to semester1-200708. However, the result that shows non-significant difference mainly involves neighbourhood heuristic that performs poorly (same as in semester1-200708 result).

Table 6.9 Semester1-200809 *p*-values comparison between *Modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 1500 iterations

Neighbourhood heuristics	Initial cost			
	18.4	15.25	12.30	9.21
Nh1	.000	.000	.000	.000
Nh2	.080	1.00	1.00	1.00
Nh3	.000	.000	.000	.000
Nh4	.000	.000	.000	.000
Nh5	1.00	1.00	1.00	1.00
Nh6	.074	.983	.452	.000
Nh7	.000	.000	.000	.000
Nh8	.000	.512	.549	.734
Nh9	.000	.000	.000	.467
Nh10	.000	.002	.844	.330

Table 6.10 Semester1-200809 *p*-values comparison between *Modified*-GDA and *Dueck*-GDA for every neighbourhood heuristics with 3000 iterations

Neighbourhood heuristics	Initial cost			
	18.4	15.25	12.30	9.21
Nh1	.000	.000	.000	.000
Nh2	.600	1.00	1.00	1.00
Nh3	.000	.000	.000	.000
Nh4	.000	.000	.000	.000
Nh5	1.00	1.00	1.00	1.00
Nh6	.035	.001	.406	.000
Nh7	.000	.000	.000	.000
Nh8	.000	.473	.108	.474
Nh9	.000	.000	.055	.144
Nh10	.000	.874	.177	.288

6.5.2.2 Comparing initial costs

We compare the initial cost based on the number of iterations for all neighbourhood heuristics for semester1-200809 dataset. As in section 6.5.1.2, we used Kruskal-Wallis to compare between the initial costs (i.e. 18.40, 15.25, 12.30 and 9.21). Referring to table 6.11, at the 95% confidence interval, there are significant differences on all of the results as the *p*-values are all less than 0.05 (reject H_0). In a pair-wise comparison between each initial cost using Mann-Whitney U, the result shows that only a few of the initial cost shows no significant differences (accept H_0) which include:

- Nh3 between 18.40 and 9.21 with 3000 iterations,
- Nh6 between 18.40 and 15.25 using 1500 iteration
- Nh8 between 15.25 and 12.30 using 3000 iteration

Based on the results, the majority of the neighbourhood heuristics show significant differences (accept H_0) and considering the result in table 6.2, it is best to start with a good quality solution and thus reaffirms our conclusions in section 6.5.1.2.

Table 6.11 Semester1-200809 *p*-value comparison for the initial cost for each neighbourhood heuristic based on the number of iterations

Neighbourhood heuristics	*p-value*	
	1500 iterations	3000 iterations
Nh1	.000	.000
Nh2	.000	.000
Nh3	.000	.000
Nh4	.000	.000
Nh5	.000	.000
Nh6	.000	.000
Nh7	.000	.000
Nh8	.000	.000
Nh9	.000	.000
Nh10	.000	.000

Table 6.12 Semester1-200809 *p*-value comparison between 1500 and 3000 iterations for each neighbourhood heuristic based on initial cost

Neighbourhood heuristics	Initial cost			
	18.40	15.25	12.30	9.21
Nh1	.000	.000	.000	.000
Nh2	.907	1.00	1.00	1.00
Nh3	.000	.000	.003	.000
Nh4	.022	.224	.124	.622
Nh5	1.00	1.00	1.00	1.00
Nh6	.000	.150	.871	.029
Nh7	.000	.000	.000	.000
Nh8	.000	.000	.644	.757
Nh9	.047	.780	.183	.322
Nh10	.024	.095	.450	.752

6.5.2.3 Comparing the number of iterations

As in 6.5.1.3, we compare the solution for the number of iterations (1500 and 3000 iterations) based on the initial cost (i.e. 1500 vs 3000 with an initial cost of 18.40, 15.25, 12.30 and 9.21). Table 6.12 shows the *p*-value of the comparison between the number of iterations. At the 95% confidence interval, the result is as follows (see table 6.12):

- Nh1, Nh3 and Nh7 show significant differences (reject H_0) across all initial costs.
- Nh2 and Nh5 show no differences (accept H_0) in the result for all initial costs.
- Nh4, Nh9 and Nh10 show significant differences (reject H_0) only on initial costs 18.40.
- Nh6 show significant difference (reject H_0) only on initial costs 18.40 and 9.21.
- Nh8 show significant difference (reject H_0) only on initial costs 18.40 and 15.25.

The results show a similar pattern as for semester1-200708 and this reaffirms our conclusion (as in the previous dataset) that is best to use a larger number of iterations.

6.5.2.4 Comparing neighbourhood heuristics

As in 6.5.1.4, we compare the set of neighbourhood heuristics based on the initial costs and the number of iterations using Kruskal-Wallis. Table 6.13 shows the p-value of the neighbourhood heuristics comparison. At the 95% confidence internal, the statistical result shows that there are significant differences (reject H_0) for the solutions produced between the neighbourhood heuristics. An in depth analysis using Mann-Whitney U shows that there are significant differences (reject H_0) for the solutions produced by most of the neighbourhood heuristics except for some. Table 6.14 summarises the significant differences (accept H_0) between neighbourhoods. Hence, we can summarise that Nh1 produced the best result, followed by Nh7 and Nh3. Next are Nh4, Nh9, Nh10, Nh8, Nh6, Nh2 and Nh5. Again, Nh1 is the best heuristic and Nh5 is the worst.

Overall, we can conclude that it is advisable to use the best quality solution as the initial solution and a larger number of iterations. In terms of neighbourhood heuristics, the results vary according to the neighbourhood heuristic and some of it performs differently between the two dataset. Hence, a neighbourhood that works for one dataset might not necessarily work on other dataset. Therefore, it is best to use a set of explorative neighbourhood heuristics (e.g. Nh1 and Nh7) as it will encourage exploration of the search space.

Table 6.13 Semester1-200809 p-value comparison for the neighbourhood heuristic based on initial cost and the number of iterations

Initial value	1500	3000
18.40	.000	.000
15.25	.000	.000
12.30	.000	.000
9.21	.000	.000

Table 6.14 Semester1-200809 summary of non-significant differences (accept H_0) when comparing neighbourhood heuristics

	1500 iterations				3000 iterations			
	18.40	15.25	12.30	9.21	18.40	15.25	12.30	9.21
Nh1	-	-	-	-	-	-	-	-
Nh2	-	Nh5	Nh5	Nh5	-	Nh5	Nh5	Nh5
Nh3	-	-	-	Nh4, Nh9	-	-	-	-
Nh4	-	-	Nh9	Nh9	-	-	Nh9	Nh9
Nh5	-	-	-	-	-	-	-	-
Nh6	Nh8, Nh9. Nh10	-	Nh10	-	Nh8, Nh10	-	Nh8	-
Nh7	-	-	-	-	-	-	-	-
Nh8	Nh9	Nh10	Nh10	-	Nh10	Nh9	Nh10	
Nh9	-	-	-	-	-	-	-	-
Nh10	-	-	-	-	-	-	-	-

'-' = result show rejecting H_0

6.6 Discussion

The proposed GDA give an improvement over the constructive heuristic and outperforms the UMP proprietary software. The success of the technique is because of its dynamic acceptance level that uses a boundary level which gradually decreases based on a decay rate, but also allows the boundary to increase when there is no improvement during search. In increasing the boundary level, the new boundary is set higher than the current solution $f(s)$ allowing the search to accept worse solutions. The algorithm also adjusts the boundary and a newly desired value is calculated when $f(s)$ is less than or equal to the desired value.

Comparison between *Modified*-GDA and *Dueck*-GDA reveal the *Modified*-GDA able to produce better solution than *Dueck*-GDA. Some of the neighbourhood heuristics do show non-significant difference. However, it mainly involves neighbourhood heuristics that perform poorly.

The *modified*-GDA gives an improvement over the initial cost (both 1500 and 3000 iterations) for the majority of the neighbourhood heuristics. Statistical analysis on the initial cost shows that some neighbourhoods (e.g. Nh3, Nh6, Nh8 etc) have similar performance, mostly between large initial costs, where semester1-200708 show more similarity compared to semester1-200809. Only a few show similarity on a small initial cost (i.e. Nh3 and Nh9), which we believe is caused by the neighbourhood heuristics themselves. The reason being, Nh3 involves moving an exam to a different timeslot only (while maintaining the selected room) and Nh9 involves swapping the exam that is chosen from amongst exam that contribute to the high value penalty. Referring to table 6.1 and table 6.2, we can summarise that using a smaller initial cost produce a higher quality solution when compared to using a larger initial cost because having a smaller initial cost encourages the search to concentrate on good regions of the search. However, note that the computational time to find a small initial cost takes a bit longer during the constructive phase (Kahar and Kendall, 2010a).

An analysis on the number of iterations, reveals that some of the neighbourhoods (i.e. Nh2, Nh5 and Nh6) show no difference in their performance between the numbers of iteration. We notice that the result is very much dependent on the heuristics used. An explorative neighbourhood would make use of the large number of iterations to efficiently explore the search space. This led us to conclude that the number of iterations does play a role in the search but it is not as important as the neighbourhood heuristics that are used. Using a larger number of iterations gives better results because it enable the method to cover more of the search space, compared to small number of iterations. However, this does require extra computational time. A good compromise is to use a small initial cost with a large number of iterations.

An analysis on the neighbourhood heuristics shows that Nh1 is the best and Nh5 is the worst. The result also show that the neighbourhood heuristics perform differently

between the two datasets (except for the first and the last two neighbourhoods), although the datasets are similar in terms of the characteristics (see chapter 3). In our observation, Nh1 (which produce the best result) is a robust neighbourhood heuristics (see table 6.1 and table 6.2). Nh2 and Nh5 are the worst neighbourhood heuristics as it is unable to improve the initial cost except for an initial cost 13.74 on semester1-200708 dataset. The result demonstrates the importance of the initial cost in order for the search to advance. Nh7 works best with a small initial cost while Nh3, Nh4 and Nh6 work best with large initial cost. Hence, we can conclude that the choice of neighbourhood heuristics is very important in the search in order to converge to a good quality solution (Thompson and Dowsland, 1998) in addition to a good choice of initial solution and number of iterations.

6.7 Contributions

The contributions of this work include an introduction of a modification of the great deluge algorithm (*modified*-GDA) that uses a simple to understand parameter that permits the boundary (that act as acceptance level) to dynamically change during the search. That is, it calculates a new boundary, decay rate and a desired value, if there is no improvement after several iterations, or, the boundary is less than the new solution, or, when the new solution is less than the desired value. We implementated the *modified*-GDA to solve the real world examination timetabling problem which includes additional constraints that have never been reported before in the literature (Kahar and Kendall, 2010a). The *modified*-GDA is able to give an improved solution over the constructive heuristic, better quality solutions compared to the proprietary software and *Dueck*-GDA approach. Finally, we investigates the effect of the initial solution, the number of iterations and neighbourhood heuristics. Statistical analysis has been carried out to determine differences between the various components. The choice of neighbourhood heuristics, number of iterations and initial solution plays a significant role in the quality of the solution returned.

6.8 Conclusion

In this work, we have investigated a real world examination timetabling problem aiming to improve on the constructive heuristic solution. The *modified*-GDA approach is able to produce good quality solutions compared to the UMP proprietary software, satisfying all the constraints (which the proprietary software fails to do), improve on the constructive result and perform better than the *Dueck*-GDA. The propose *modified*-GDA uses a simple to determine parameter that can find a good solution. The selection of neighbourhood heuristics, iterations and initial cost plays a significant part in the search.

Due to the fact that the neighbourhood heuristics are very important, we are going to investigate the use of multiple neighbourhood. We are going to use each neighbourhood in succession. The next neighbourhood will be selected if the current neighbourhoods show no improvement. This will be discussed further in the next chapter.

Chapter 7

Solving a Real World Examination Timetabling Problem: Multi-Neighbourhood Great Deluge Algorithm

Many search methods can be found in the scientific literature, with meta-heuristics being very popular. Meta-heuristics are very dependent on parameter settings and the neighbourhoods used in order to find good quality solutions (Burke and Newall, 2002 and Burke and Petrovic, 2002). This creates a problem for exam timetable officers where it is often difficult to determine the best parameter setting and neighbourhood heuristics to guarantee a good quality solution (Thompson and Dowsland,1996).. Therefore it is up to the algorithm designer to automate this process as far as possible. This work introduces a modified extended Great Deluge Algorithm with multi-neighbourhood heuristics for the examination timetabling problem, which uses a single, easy to understand parameter and calls upon more than one neighbourhood during the search. We investigate different ordering strategies, as well removing several of the good and worse neighbourhood heuristics in order to study the effect. Statistical analysis is carried out to compare the results between different strategies. The proposed methodology is able to produce good quality solutions when compared to the solution currently produced by the host organisation and also when compared to the solutions generated in our previous work.

In section 7.1, gives an introduction of the work presented in this chapter. We describe the modified GDA using multi-neighbourhood heuristics in sections 7.2. The experimental setup is discussed in section 7.3. The results from the improvement phase is shown in section 7.4 and, in section 7.5, we analyse the results via a set of statistical tests. Discussion of the result and statistical analysis is presented in 7.6. Lastly, in section 7.7 and 7.8, we summarise the contribution and present our conclusions.

7.1 Introduction

There are many search methodologies that can be used to generate examination timetables. One class in particular are meta-heuristic approaches. Meta-heuristics tend to be very dependent on parameter settings (Petrovic and Burke, 2004) and the neighbourhood operators that are used (Ahuja, Orlin and Sharma, 2000; Kahar and Kendall, 2010b and Thompson and Downsland, 1998). Each neighbourhood operator affects the solution in a different way (Ahuja, Orlin and Sharma, 2000). A suitable neighbourhood operator for one dataset might not perform well for another (Kahar and Kendall, 2010b).

We propose a modification of the great deluge algorithm (GDA) proposed by Dueck (1993) which uses a simple to understand parameter with a dynamic boundary level (acceptance level) that changes during the search. Additionally, the proposed method uses more than one neighbourhood heuristic during the search. This allows the search to explore a wider range of possibilities in the search space. We investigate the proposed methodology on a real world examination timetabling problem from UMP. This work is an extension of our previous work, where we developed a constructive heuristic for this real world problem (Kahar and Kendall, 2010a) and improved on that solution using single neighbourhoods and also explored if the number of iterations and the starting solution led to statistically different results (Kahar and Kendall, 2010b). In this work we are investigating whether providing GDA with a set of neighbourhood moves from which to choose can further improve the algorithm.

7.2 Modified Great Deluge Algorithm

A suitable parameter setting is important in meta-heuristics and it is often difficult to determine the best value to guarantee a good quality solution (Petrovic and Burke, 2004). In addition, meta-heuristic techniques are often dependent on neighbourhood heuristics to effectively explore the search space. Different neighbourhood heuristics could produce a different solution within the same search space (Ahuja, Orlin and Sharma, 2000; Kahar and Kendall, 2010b). In our previous work (Kahar and Kendall, 2010b), which considered two different UMP datasets that were almost identical (in terms of the conflict density matrix, see table 3.2), we showed that one single neighbourhood did not always produce the best result and this motivated this study to make a set of neighbourhoods available. Furthermore, having simple and easy to understand parameters (i.e. computational time and desired value) to determine the decay rate in Burke et al. (2004) makes it straightforward for non-experts (e.g. university timetable officers) to set the parameters, especially when compared to other meta-heuristic techniques (e.g. SA, TS, GA etc).

The success of GDA and the simplicity in parameter setting, motivates us to explore this method with the aim of bringing the modified multi-neighbourhood GDA to the university timetable officer as they are the ones responsible for producing the timetable at UMP. This work is an extension of our previous work in Kahar and Kendall (2010b), exploring the use of simple parameter settings together with multi-neighbourhood heuristics (the algorithm uses more than one neighbourhood heuristic during the search). The use of multi-neighbourhood removes the needs to make algorithmic choices (i.e. choosing the neighbourhoods) which they (exam timetabling officers) are probably not in a position to do effectively. The following neighbourhood heuristics are used in our experiments. Note that, unless stated otherwise all the exam, timeslot and rooms are selected randomly. The exact same neighbourhood heuristics as in section 6.3 are used here.

The algorithm works by using the current neighbourhood in every iteration and only selects a different neighbourhood (within the list) when the current neighbourhood solution is rejected by the GDA (solution is greater than the boundary level). The

neighbourhood heuristics are sorted randomly and are also based on the result in Kahar and Kendall (2010b). In Kahar and Kendall (2010b), the most effective neighbourhoods for semester1-200708 were Nh1 follow by Nh7, Nh4, Nh3, Nh9, Nh6, Nh8, Nh10, Nh2 and Nh5. In semester1-200809 the most effective neighbourhoods were found to be Nh1 followed by Nh7, Nh3, Nh4, Nh9, Nh10, Nh8, Nh6, Nh2 and Nh5. In the discussion that follows we refer to these as the *specified* neighbourhoods. Our proposed modified multi-neighbourhood GDA is shown in figure 7.1.

1. *Set the initial solution s from the constructive heuristic (Kahar and Kendall, 2010a);*
2. *n is the neighbourhood heuristics N, where* $n \in \{1...N\}$
3. *Calculate initial cost function* $f(s)$
4. *Set the desired value* $\boldsymbol{D}$
5. *Set the number of iterations* $\boldsymbol{I}$
6. *Set Initial Boundary Level* $\boldsymbol{B} = 0.03f(s) + f(s)$
7. *Set initial decay Rate* $\Delta \boldsymbol{B} = (B - D)/I$
8. *Set* $s_{best} = s$
9. *Sort N randomly or according to a specified sequence*
10. *While stopping criteria not met do*
11. *Apply neighbourhood heuristic n on s to obtain* s^*
12. *Calculate* $f(s^*)$
13. *If* $f(s^*) \leq f(s)$ *or* $f(s^*) \leq B$ *then*
14. *Accept* $s = s^*$
15. *If* $f(s^*) \leq f(s_{best})$ *then*
16. $s_{best} = s^*$
17. *If* $f(s^*) = f(s)$ *then*
18. $n = n + 1$
19. *Else*
20. $n = n + 1$
21. *If* $n > N$ *then*
22. $n = 1$
23. *Lower Boundary* $\boldsymbol{B} = \boldsymbol{B} - \Delta \boldsymbol{B}$
24. *If no improvement in iterations W or* $B \leq f(s_{best})$ *or* $f(s) \leq D$ *then*
25. *Set* $s = s_{best}$
26. *If* $f(s) \leq D$ *then*
27. $D = f(s)*0.8$
28. *Set new decay rate* $\Delta \boldsymbol{B} = (f(s) - D)/I_{remaining}$
29. *Set* $\boldsymbol{B} = 0.03f(s) + f(s)$

Figure 7.1 Our proposed Great Deluge algorithm

The algorithm starts by calculating the intital cost function $f(s)$ (lines 1-3). Next, we set the desired value D, number of iterations I and the boundary level B (lines 4-6). The boundary level B is set 3% higher than the initial solution $f(s)$ obtained from a constructive heuristic (Kahar and Kendall, 2010a). The boundary level B is increased

slightly to allow acceptance of worse solutions. The decay rate ΔB is calculated as the difference between boundary level B and the desired solution D divided by the number of iterations I (line 7). Based on the decay rate formulation, obviously having a small number of iterations would result in a steeper rate compared to using a larger number of iterations.

Next, we sort the neighbourhood heuristics N randomly or according to a specified sequence based on a work in Kahar and Kendall (2010b). While the stopping condition is not met, we apply neighbourhood heuristic n to the current solution s (line 11). We calculate the new cost value $f(s^*)$ where $s^* \in N(s)$ (line 12). s^* is accepted if $f(s^*)$ is less than $f(s)$ or if $f(s^*)$ less than boundary B (lines 13-14). Next, If $f(s^*)$ is less than $f(s_{best})$, set $s_{best} = s^*$ (line 15-16). Then, if $f(s^*)$ is equal to $f(s)$, we select the next neighbourhood n from the neighbourhood list ($n=n+1$, line 17-18). However, if s^* is not accepted, select the next neighbourhood n ($n=n+1$, line 19-20). In a condition where n is greater than N, we set $n = 1$ (line 21-22). Next, boundary B is lowered based on the decay rate, ΔB (line 23). However, if there is no improvement for several iterations, W ($W = 20$ in this work) or boundary B is less than or equal to $f(s_{best})$ or $f(s)$ is less than or equal to desired value D; then set $s = s_{best}$ (line 25). The new decay rate ΔB is calculated as the difference between $f(s)$ and desired value D divided by the remaining number of iterations I (line 28). However, if $f(s)$ is less than, or equal to, the desired value D then a new desired value is calculated as 80% of $f(s)$ (line 26-27). Additionally, the boundary is set slightly above $f(s)$ (line 29).

7.3 Experimental setup

We implemented the propose method to two of the UMP datasets. A details discussion of the dataset refer to chapter 3. The same properties as in Kahar and Kendall (2010b) are used here to allow comparison between these methods. In this experiment, we use an initial solution of 7.82 for semester1-200708 and 9.21 for semester1-200809. These solution are created by the constructive heuristic in Kahar and Kendall, (2010a). Each experiment was run 50 times on a Pentium core2 processor. We ran for 3000 iterations (≈ 960 seconds).

As described above, the sorting used is a *random* and *specified neighbourhood* sorting strategy. Additionally, we also experimented by removing the good and the worst three neighbourhoods. In semester1-200708, the good neighbourhoods are the first three (i.e. Nh1, Nh7 and Nh4), and the worse are the last three (i.e. Nh10, Nh2 and Nh5). In semester1-200809, the good neighbourhoods are Nh1, Nh7 and Nh3 and the worse are Nh6, Nh2 and Nh5. We will carry out an experiment to determine if there is any significant difference in removing these neighbourhoods which would mean having to implement a smaller number of neighbourhoods which might be attractive to some developers.

7.4 Examination assignment: Results

In this section, we compare the examination timetable generated by the UMP proprietary software, the constructive heuristic (Kahar and Kendall, 2010a), the modified-GDA (Kahar and Kendall, 2010b) and our modified multi-neighbourhoods GDA. The result for semester1-200708 is shown in table 7.2 and semester1-200809 is shown in table 7.4.

7.4.1 Semester1-200708

The UMP result generated by the proprietary software for semester1-200708 is 13.16 with a violation of one of the hard constraints (no clashing constraint - Kahar and Kendall, 2010a). Using the constructive heuristic (Kahar and Kendall, 2010a), we manage to construct a feasible solution using different candidate list sizes (C = 1 and C = 5). With C = 1, the minimum value produced is 10.98 while C = 5 produced a solution with an objective function of 4.74. In the improvement phase, the modified GDA was able to give an improved solution on this initial solution. We experiment with different initial solutions (Kahar and Kendall, 2010b) and manage to produce a minimum value of 4.01, starting with an initial cost of 7.82. Table 7.1 summaries the best results for semester1-200708 using different techniques (including the methodology proposed here).

Referring to table 7.2, the modified multi-neighbourhood GDA (*MuNeiGDA*), with *random* ordering strategies give a minimum value of 3.30 with an average of 3.89. The *specified* ordering strategies gives a minimum value of 3.41 with an average of 3.86. Note that the initial cost used is 7.82. Compared to the initial solution used (7.82), the *MuNeiGDA-random* ordering is able to produce a solution that is 58% (7.82 compared with 3.30 ((7.82 - 3.30)/7.82 x 100%)) better and in *MuNeiGDA-specified*, it is also able to produce an improvement of 56% (7.82 compared with 3.41 ((7.82 - 3.41)/7.82 x 100%)).

Table 7.1 Summary results for semester1-200708

Techniques	Ave	Stdev	Min	Max
Constructive heuristic ($C = 1$)	15.51	2.10	10.98	20.03
Constructive heuristic ($C = 5$)	6.06	0.76	4.74	7.98
Modified GDA	4.38	0.15	4.01	4.73
MuNeiGDA-Random	3.89	0.18	**3.30**	4.23
MuNeiGDA-Specified	3.86	0.16	3.41	4.19

Using a *random* ordering strategy (*MuNeiGDA-random*); we are able to produce solution that is 75% (13.16 compared with 3.30 ((13.16 - 3.30)/13.16 x 100%)) better when compared to the solution produced by the UMP proprietary software. The *MuNeiGDA-random* also outperforms candidate list, $C = 5$, by 30% (4.74 compared with 3.30 ((4.74 - 3.30)/4.74 x 100%)) and outperforms the *modified*-GDA (Kahar and Kendall, 2010b) by 18% (4.01 compared with 3.30 ((4.01 - 3.30)/4.01 x 100%)).

In the *specified* ordering strategies (*MuNeiGDA- specified*); we are able to produce a solution that is 74% (13.16 compared with 3.41 ((13.16 - 3.41)/13.16 x 100%)) better when compared to the solution produced by the UMP proprietary software. The *MuNeiGDA-specified* also outperforms the candidate list, $C = 5$, by 28% (4.74 compared with 3.41 ((4.74 - 3.41)/4.74 x 100%)) as well as the modified GDA (Kahar and Kendall, 2010b) by 15% (4.01 compared with 3.41 ((4.01 - 3.41)/4.01 x 100%)). Table 7.1 summaries the result for semester1-200708.

Overall the proposed *MuNeiGDA* algorithm gives an improvement when compared to the UMP proprietary software (Kahar and Kendall, 2010a) and the modified-GDA (Kahar and Kendall, 2010b).

Table 7.2 GDA with multi neighbourhood result for semester1-200708 based on random and specified neighbourhood ordering strategies

Multi neighbourhood	Random				Specified			
	Std	Ave	Min	Max	Std	Ave	Min	Max
All Nh	0.24	4.01	3.54	4.59	0.25	3.90	3.46	4.59
Remove Nh5	0.21	3.93	3.36	4.42	0.19	3.88	3.44	4.50
Remove Nh2	0.18	**3.89**	**3.30**	4.23	0.16	3.86	**3.41**	4.19
Remove Nh10	0.25	3.99	3.50	4.61	0.20	3.89	3.47	4.35
Remove Nh5 and Nh2	0.22	4.04	3.63	4.53	0.19	4.01	3.66	4.66
Remove Nh5 and Nh10	0.18	3.91	3.49	4.27	0.18	**3.85**	3.50	4.28
Remove Nh2 and Nh10	0.19	3.89	3.51	4.33	0.15	3.90	3.50	4.20
Remove Nh5, Nh2 and Nh10	0.19	4.04	3.58	4.40	0.23	4.06	3.54	4.49
Remove Nh1	0.23	4.07	3.60	4.61	0.17	4.00	3.67	4.46
Remove Nh4	0.24	4.11	3.60	4.61	0.22	3.97	3.48	4.40
Remove Nh7	0.22	4.04	3.60	4.45	0.21	3.97	3.55	4.56
Remove Nh1 and Nh4	0.25	4.29	3.75	4.80	0.24	4.21	3.74	5.04
Remove Nh1 and Nh7	0.20	4.16	3.64	4.56	0.21	4.07	3.65	4.51
Remove Nh4 and Nh7	0.23	4.21	3.64	4.64	0.17	3.99	3.54	4.46
Remove Nh1, Nh4 and Nh7	0.26	4.47	3.96	5.18	0.22	4.31	3.75	4.87

Std = stdev; Ave = average; Min = Minimum; Max = Maximum

7.4.2 Semester1-200809

In semester1-200809, the calculated UMP solution was 26.08 with a violation of all of the hard constraints (Kahar and Kendall, 2010a). In the constructive heuristic (Kahar and Kendall, 2010a), a candidate list of $C = 1$ produced a minimum value of 13.89 and with a candidate list of $C = 5$ the result achieved was 6.61. During the improvement phase, the modified-GDA was able to improve on the initial solution used. We experiment with different initial solutions (Kahar and Kendall, 2010b) and manage to produce a minimum value of 5.63 using an initial cost of 9.21. Table 7.3 summaries these results using different techniques.

Referring to table 7.3, the *MuNeiGDA-random* ordering strategies produce a minimum value of 5.52 with an average of 6.09. The *specified* ordering strategies produces a cost value which is almost the same as the *random* ordering, that is 5.53 with an average of 5.99. Note that the initial cost used is 9.21. Compared to the initial solution used (9.21), the *MuNeiGDA-random* and *MuNeiGDA-specified* gives an improvement of 40% (9.21 compared with 5.52 ((9.21 – 5.52)/9.21 x 100%)). The *MuNeiGDA-random* produced a solution that is 79% (26.08 compared with 5.52 ((26.08 – 5.52)/26.08 x 100%)) better when compared to the solution produced by the UMP proprietary software and 16% (6.61 compared with 5.52 ((6.61 – 5.52)/6.61 x 100%)) better when compared to candidate list, $C = 5$. Compared with the modified-GDA, the *MuNeiGDA-random* outperforms the modified-GDA (Kahar and Kendall, 2010b) by only 2% (5.63 compared with 5.52 ((5.63 – 5.52)/5.63 x 100%)). The same performance was also shown in the *MuNeiGDA-specified.* A good average result shown only when removing Nh2 or Nh5.

Table 7.3 Summary results for semester1-200809

	Ave	**Stdev**	**Min**	**Max**
Constructive heuristic (C = 1)	17.33	1.69	13.89	21.66
Constructive heuristic (C = 5)	7.88	0.71	6.61	9.69
Modified GDA	6.04	0.15	5.63	6.42
GDA-MuNei (Random)	6.09	0.20	**5.52**	6.48
GDA-MuNei (specified)	5.99	0.19	5.53	6.47

The result for semester1-200809 as described above is shown in table 7.4. Overall the proposed *MuNeiGDA* gives an improvement when compared to the UMP proprietary software and the modified-GDA. In the next section, we are going to further analysis the results using statistical comparison.

Table 7.4 GDA with multi-neighbourhood result for semester1-200809 based on the *random* and specified neighbourhood ordering strategies

Multi neighbourhood	Random				Specified			
	Std	Ave	Min	Max	Std	Ave	Min	Max
All Nh	0.22	6.10	5.58	6.67	0.15	6.06	5.71	6.40
Remove Nh5	0.17	5.98	5.56	6.36	0.16	5.97	5.55	6.22
Remove Nh2	0.18	6.01	5.66	6.38	0.19	5.99	**5.53**	6.47
Remove Nh6	0.20	6.16	5.70	6.52	0.16	6.01	5.73	6.38
Remove Nh5 and Nh2	0.14	6.03	5.68	6.28	0.19	6.07	5.60	6.53
Remove Nh5 and Nh6	0.19	6.04	5.68	6.41	0.22	6.03	5.58	6.49
Remove Nh2 and Nh6	0.20	6.09	**5.52**	6.48	0.17	6.02	5.58	6.33
Remove Nh5, Nh2 and Nh6	0.16	6.05	5.67	6.34	0.16	6.08	5.75	6.43
Remove Nh1	0.20	6.28	5.76	6.67	0.22	6.21	5.75	6.69
Remove Nh3	0.24	6.29	5.60	6.72	0.19	6.16	5.78	6.51
Remove Nh7	0.18	6.22	5.80	6.65	0.17	6.12	5.84	6.51
Remove Nh1 and Nh3	0.19	6.56	6.24	6.94	0.18	6.49	5.95	6.91
Remove Nh1 and Nh7	0.18	6.51	6.08	6.95	0.19	6.46	6.08	6.85
Remove Nh3 and Nh7	0.19	6.42	5.92	6.78	0.17	6.29	5.94	6.71
Remove Nh1, Nh3 and Nh7	0.19	7.04	6.68	7.49	0.20	7.07	6.63	7.59

Std = stdev; Ave = average; Min = Minimum; Max = Maximum

7.5 Statistical Comparisons

This section present the statistical analysis carried out on our results. The aim is to compare strategies used in the experiments and determine whether there are statistical differences. The comparisons include:

a) Comparison between different ordering strategies: Is there any significant difference in using *random* ordering compared to *specified* neighbourhood ordering strategies?

b) Comparison between different sets of neighbourhood heuristics: Is there any significant difference in using all of the neighbourhoods compared to removing three of the good or worst neighbourhoods?

All data was tested for normality using Shapiro-Wilk with H_0 - assumes that the sample is normally distributed, and H_1 - assumes that the sample is non-normal. We reject H_0 when $p \leq 0.05$ and vice versa. In fact, all data is normally distributed,

therefore we use *t*-test and one-way ANOVA, followed by Games Howell post-hoc to determine any significant differences.

The hypotheses for the *t*-test and one-way ANOVA are that the null hypothesis H_0 - assumes that the samples are from identical populations, and the alternative hypothesis H_1 - assumes that the sample comes from different population. We reject H_0 when $p \leq 0.05$ and vice versa. The above hypothesis are used throughout the statistical tests described in the following sections. The *t*-test is used to compare two samples while one-way ANOVA is used to compare more than two samples. Additionally, Games Howell Post-Hoc is used in conjunction with one-way ANOVA to investigate the cause of H_0 rejection. Games Howell Post Hoc compares more than one pair of samples simultaneously.

7.5.1 Semester1-200708

7.5.1.1 Ordering strategies

In a comparison on the ordering strategies, we want to test whether there is any significant difference in the performance on *random* ordering compare to using *specified neighbourhood* ordering strategies with the *MuNeiGDA*. *T*-test is used in the statistical test, the result shows a significant difference (reject H_0) between the two ordering strategies when using all of the neighbourhoods. Referring to table 7.5, in removing the worst neighbourhood heuristics (Nh5; Nh2; Nh10; Nh5 and Nh2, etc), the solutions show no significant difference (accept H_0) except when removing Nh10 (reject H_0). However when removing good neighbourhoods (Nh1; Nh4; Nh7; Nh1 and Nh4, etc), the result shows:

- Significant differences (reject H_0) when removing Nh4; Nh1 and Nh7; Nh4 and Nh7; and, Nh1, Nh4 and Nh7.
- No significant difference (accept H_0) when removing Nh1; Nh7 and, Nh1 and Nh4.

Table 7.5 *p*-value result for semester1-200708 in comparison between the ordering strategies

Multi neighbourhood	*T*	*Df*	*p*
All Nh	2.23	98.00	0.03
Remove Nh5	1.29	98.00	0.20
Remove Nh2	0.76	98.00	0.45
Remove Nh10	2.20	98.00	0.03
Remove Nh5 and Nh2	0.63	98.00	0.53
Remove Nh5 and Nh10	1.58	98.00	0.12
Remove Nh2 and Nh10	-0.17	98.00	0.87
Remove Nh5, Nh2 and Nh10	-0.57	98.00	0.57
Remove Nh1	1.87	98.00	0.07
Remove Nh4	3.01	98.00	0.00
Remove Nh7	1.82	98.00	0.07
Remove Nh1 and Nh4	1.64	98.00	0.11
Remove Nh1 and Nh7	2.29	98.00	0.02
Remove Nh4 and Nh7	5.28	98.00	0.00
Remove Nh1, Nh4 and Nh7	3.19	98.00	0.00

Based on the result, we notice that in removing the worst neighbourhood, it does not show any significant difference (accept H_0) between the ordering strategies when removing one or more of the worst neighbourhood(s). However, in removing the good neighbourhood(s), overall it shows a significant difference (reject H_0) between the ordering strategies particularly when more than one neighbourhood is removed.

We can conclude that they are no differences to the ordering strategies when removing the worst neighbourhoods. The result mainly shows significant differences only when removing the good neighbourhoods, especially when removing more than one neighbour. Hence, based on the result shown in table 7.2 and table 7.5, it is best to use a specified ordering strategy.

7.5.1.2 Neighbourhood heuristics used

In this statistical test, we compare the use of neighbourhood heuristics to determine whether is there a significant difference in using all of the neighbourhood heuristics

compared to removing the worst or removing the good neighbourhoods (based on previous work in Kahar and Kendall, 2010b). One-way ANOVA is used to compare between these neighbourhood heuristics. Based on the statistical test, the result show significant differences (reject H_0) with p-values = 0.00.

In a pair-wise comparison using Games Howell Post-Hoc on the *random* and *specified* ordering strategies, the result shows significant difference (reject H_0) when we remove the good neighbourhoods (see table 7.7a and table 7.7b, appendix K), particularly when more than one good neighbourhood is removed (see column l - o in table 7.7a, 7.7b, 7.8a and 7.8b; appendix K). Therefore, we can conclude that by discarding more than one good neighbourhood will lead to a deterioration of algorithmic performance.

7.5.2 Semester1-200809

7.5.2.1 Ordering strategies

In a comparison on the ordering strategies for semester1-200809, the result shows a significant difference (*p-values* = 0.41) between *random* and *specified* neighbourhood ordering strategies when using all of the neighbourhoods (see table 7.6). In removing the worst three neighbourhood heuristics (Nh5; Nh2; Nh6; Nh5 and Nh2, etc), the solutions show no significant difference (accept H_0) except when removing Nh6 (see table 7.6). In removing the *specified* (good) neighbourhoods (Nh1; Nh3; Nh7; Nh1 and Nh3, etc), the result shows:

- Significant differences (reject H_0) when we remove Nh1; Nh1 and Nh3; Nh1 and Nh7; and Nh1, Nh3 and Nh7.
- No significant differences (accept H_0) when removing Nh3; Nh7; and Nh3 and Nh7.

Therefore, we can conclude that, in removing the worst neighbourhood, there are no differences between *random* and *specified* ordering when removing either one or more of the worst neighbourhood(s). However, the result show non significant differences (accept H_0) between the ordering strategies when we remove the good

neighbourhoods. Hence, based on the result shown in table 7.5 and table 7.6, it is best to use a *specified* ordering strategy.

Table 7.6 *p*-value result for semester1-200809 in comparison between the ordering strategies

Multi neighbourhood	*t*	*df*	*P*
All Nh	0.83	98.00	0.41
Remove Nh5	0.30	98.00	0.76
Remove Nh2	0.78	98.00	0.44
Remove Nh6	4.30	98.00	0.00
Remove Nh5 and Nh2	-0.98	98.00	0.33
Remove Nh5 and Nh6	0.26	98.00	0.80
Remove Nh2 and Nh6	1.85	98.00	0.07
Remove Nh5, Nh2 and Nh6	-1.00	98.00	0.32
Remove Nh1	1.67	98.00	0.10
Remove Nh3	3.01	98.00	0.00
Remove Nh7	2.87	98.00	0.01
Remove Nh1 and Nh3	1.87	98.00	0.07
Remove Nh1 and Nh7	1.14	98.00	0.26
Remove Nh3 and Nh7	3.49	98.00	0.00
Remove Nh1, Nh3 and Nh7	-0.81	98.00	0.42

7.5.2.2 Neighbourhood heuristics used

In a comparison between the neighbourhood heuristics using one-way ANOVA, the statistical test shows that there are significant differences (reject H_0) on all of the results with *p*-values = 0.00. In a pair-wise comparison using Games Howell Post-Hoc on the *random* and *specified* ordering strategies, the result shows that only good neighbourhood heuristics show significant differences (reject H_0) when it is removed (see table 7.8a and table 7.8b, appendix K) mainly when more than one good neighbourhood is removed (see column l to o in table 7.8, appendix K). As in semester1-200708, we can conclude that the algorithm will not work effectively when the good neighbourhoods are removed.

7.6 Discussion

The proposed modified multi-neighbourhoods GDA is able to give a better cost value compare to the *modified*-GDA (Kahar and Kendall 2010b) and outperforms the UMP proprietary software. The *multi-neighbourhoods* GDA is able to produce a better solution because of the use of a dynamic acceptance level and having the benefit of more than one neighbourhood.

The dynamic acceptance level approach uses, a boundary level which gradually decreases the decay rate, but also allows the boundary to increase when there is no improvement during the search (for several iterations). In increasing the boundary level, the new boundary is set higher than the current solution $f(s)$ allowing the search to accept worse solutions (encouraging exploration). Additionally, the algorithm adjusts the boundary when it is less than $f(s_{best})$ and even when $f(s)$ is less than or equal to the desired value, D. However for the latter condition, the algorithm will calculate a new desired value, D.

In addition, having the multi-neighbourhood heuristics feature increases exploration of the search space due to the fact that different neighbourhood heuristics perform differently (Kahar and Kendall, 2010b). The multi-neighbourhood uses the current neighbourhood as long as the result is accepted and only selects a different neighbourhood when the current result shows no improvement (compared to the boundary level).

Overall, the proposed *MuNeiGDA* (*random* and *specified* ordering) approaches gives an improvement when compared to the UMP proprietary software (Kahar and Kendall, 2010a) and the *modified*-GDA (Kahar and Kendall, 2010b). Referring to the ordering strategies result, the *MuNeiGDA*-random ordering strategies produces a better minimum (*min*) value when compared to *specified* ordering but on average the *random* ordering gives a slightly higher value for both of the datasets. Additionally, in the statistical tests, it shows no significance difference (accept H_0) in using either *random* or *specified* ordering (considering the choice of neighbourhood that give the *min*

value). Hence, we conclude that the ordering strategies give a minimum impact towards the algorithm's performance as both orderings perform the same.

In a comparison on the neighbourhood heuristics used, both of the datasets show significant differences with p-values < 0.05 (reject H_0). A detailed comparison using Games Howell Post-Hoc reveal that (for both datasets), only good neighbourhood heuristics show significant differences (reject H_0) when they are removed, particularly when we remove more than one good neighbourhood. Therefore, referring appendix I, it is best to use all of the neighbourhood as the solution shows small differences compared to removing the worst neighbourhoods. Hence, conclude that it is advisable to use all of the neighbourhood heuristics as each of the neighbourhoods have their own strengths that could aid the algorithm in exploring the search space.

7.7 Contributions

The contributions are as follows:

a) Present a modification of the great deluge algorithm (GDA) that uses a simple to understood parameter and permits the boundary (that acts as an acceptance level) to dynamically change during the search. It is dynamic in the sense that it calculates a new boundary, decay rate and a desired value, if there is no improvement after several iterations, or, the boundary is less than the new solution, or, when the new solution is less than the desired value. The algorithm uses more than one neighbourhood heuristic (multi-neighbourhood) during the search. It will use the current neighbourhood heuristic until the result shows no improvement and then it will choose the next neighbourhood.
b) We have explored suitable ordering strategies and neighbourhood heuristics by removing the worst or best neighbourhoods. Statistical test were carried out to determine the statistical differences between the ordering and the choice of neighbourhood heuristics. This revealed that it is best to use all of the neighbourhoods as this helps to better explore the search space.

c) Implementation of (a modified multi-neighbourhood) GDA in solving a real world examination timetabling problem which includes constraints that never been reported before in the scientific literature (Kahar and Kendall, 2010a). The multi-neighbourhood GDA is able to give better quality solutions compared to the proprietary software and the original modified-GDA.

7.8 Conclusion

In this chapter, we have investigated a real world examination timetabling problem aiming to improve the constructive heuristic solution. The modified multi-neighbourhood GDA approach is able to produce good quality solutions compared to the UMP proprietary software, satisfying all the constraints (which the proprietary software fails to do) and improve on the constructive result. The propose GDA uses a simple to determine parameter that can find a good solution and is able to find a better solution than the initial cost even with higher desired value (due to it capabilities to adjust the desired value, boundary and decay rate). Having a simple and versatile algorithm helps to eliminate the difficulty for the examination timetabling officer in managing/using the algorithm.

Additionally, the use of multi-neighbourhood heuristics help to effectively explore the search space and improve on the result. The multi-neighbourhood simplifies the operation of the algorithm for the timetabling officer rather than having to determine the suitable neighbourhood. This is beneficial as in Kahar and Kendall (2010b) they showed that the choice of neighbourhood plays a major role in a search. Lastly, we conclude that, the multi-neighbourhood successfully shows that it is able to provide a better solution compare to the modified-GDA. It is best to use all of the neighbourhood heuristics rather than having to select a suitable set of neighbourhoods.

Chapter 8

Solving ITC2007 Examination Timetabling Problems

The Second International Timetabling competition (ITC2007) introduced with the aim of creating a platform for researchers to test their algorithms on real world timetabling problems. It includes more realistic problems and contains comprehensive constraints compared to other benchmark examination dataset in the literature. In this chapter, we report the implementation of graph heuristics, *modified*-GDA and *multi-neighbourhood* GDA to the ITC2007 examination dataset. The aim of the experiment is to determine whether the above methods able to solve the ITC2007 as it did for the UMP datasets.

The chapter is organised as follows, in sections 8.1, we describe the ITC2007 examination problem. In sections 8.2, we describe the experimental setup to allow reproducibility for other researchers. The result of the proposed method is shown in section 8.3. Discussion on the results is presented in section 8.4. Lastly, in sections 8.5 and 8.6 we summarise the contributions and present our conclusions.

8.1 International Timetabling Competition 2007 (ITC2007)

The First International Timetabling competition was established in 2002 with the aim of introducing a real world timetabling problems for researchers to test their

algorithms. Recently, the international timetabling competition 2007 (ITC2007) which is the second competition series has been established to further attract and bring the researcher together in exploring the timetabling problem. The second competition (called ITC2007) include examination problem from the third track (http://www.cs.qub.ac.uk/itc2007/index) where it has eight different dataset each with different features i.e. number of exam, number of timeslot and rooms etc. (see section 2.4.4, table 2.6). The datasets vary in their level of conflict density and constraints (see table 8.1). Beside the conflict density, the difficulty level dependent on the constraints and these includes number of timeslot, number of room, room capacity, timeslot length, period hard constraints (PHC) and room hard constraints (RHC). Detail descriptions of these constraints are as follows:

a) ***Number of timeslot:*** the number of timeslot differs between the datasets. There are penalty associated with the timeslot that is two exams in a row (*S1*) or day (*S2*), spreading (*S3*), later period (*S5*) and period penalty (*S6*) (see figure 8.2).
b) ***Number of rooms:*** the number of rooms differs between the datasets. Having large room quantity give flexibility in choosing the room. There are penalty associated with a certain room (see *S7* in figure 8.2)
c) ***Room capacity***: ITC2007 datasets allow the exam to share room but disallow the exam to be split into several rooms.
d) ***Timeslot length:*** some of the datasets have a similar and different timeslot length (see table 8.1). A different timeslot length would require a check for suitable timeslot length during *exam-timeslot* assignment. There also a penalty associated with different length of exams sharing the same room (see *S4* in figure 8.2).
e) ***Period hard constraints - AFTER:*** the AFTER constraint involve schedule the second exam AFTER the first exam timeslot. Example schedule *examA* AFTER *examB*. *ExamA* need to be schedule in timeslot after *examB* timeslot. Some of the dataset include more than two exams that associates with AFTER constraint (e.g. *examA* AFTER *examB* and *examA* AFTER *examC*) and this complicates the problem further (see Exam-1 and Exam-8 in table 8.1). Furthermore, having a multi period hard constraint (e.g. *examA* AFTER *examB*

and *examA* coincidence *examC*) complicates the problem even further (see Exam-5, Exam-7 and Exam-8 in table 8.1).

f) ***Period hard constraints - EXCLUSION***: the exclusion constraint involves scheduling the exam in a different timeslot to one another (e.g. examA EXCLUSION examB).

g) ***Period hard constraint - COINCIDENCE***: the coincidence constraint involves scheduling the exam in the same timeslot (e.g. *examA* COINCIDENCE *examB*).

h) ***Room hard constraints***: this constraint require exam to be schedule into the allocate room. Only 3 datasets contains this constraint that is Exam-2, Exam-3 and Exam-8. Table 8.1 shows the number of exams that involve with the constraints.

Table 8.1 ITC2007 examination datasets features

	Timeslot length	Period Hard Constraint (PHC)			Room Hard Constraint (RHC)
		After (*Af*)	Exclusion (*Ec*)	Coincidence (*Cd*)	
Exam-1	Similar	max-3	max-2	max-2	-
Exam-2	Vary	max-2	max-2	max-2	2
Exam-3	Vary	max-2	max-2	max-4	15
Exam-4	Similar	No	max-4	max-2	-
Exam-5	Vary	max-2^{Cd}	max-2^{Cd}	max-$4^{Af,Ec}$	-
Exam-6	Vary	max-2	max-2	max-2	-
Exam-7	Similar	max-2^{Ec}	max-3^{Af}	max-2	-
Exam-8	Similar	max-5^{Cd}	-	max-5^{Af}	1

**similar = the timeslot length are the same for all timeslots; vary = the timeslot differ in length among them; max-3^{xx}= the maximum number of exam involve with the respectively period hard constraint (e.g. examA AFTER examB and examA AFTER examC) while the xx referred to the exam(s) involve in multi period hard constraint (e.g. examA AFTER examB and examA COINCIDENCE examC)*

Figure 8.1 and figure 8.2 shows the hard and soft constraints for the ITC2007 examination datasets. The hard constraints need to be satisfied for a feasible solution. The soft constraints need to be satisfied as much as possible and, hence it is used to determine the quality of the solution.

Figure 8.1 Hard Constraints

H1.	Student cannot sits more than one exam at the same time
H2.	The exams capacity should not exceed the room capacity
H3.	The exam length should not violate the timeslot lengths
H4.	A sequence or ordering of an exams must be respected, e.g. schedule *ExamA* after *ExamB*;
H5.	Schedule exam into specified room (room related hard constraints) e.g. *ExamA* must schedule to Room 11

Figure 8.2 Soft Constraints

S1.	*Two exams in a row*: minimise student sitting consecutive exams on the same day.
S2.	*Two exams in a day*: minimise student sitting more than two exams in a day (only applied if more than two (2) timeslot per day).
S3.	*Spreading of exam*: Each set of student examinations should be spread as evenly as possible over the exam period.
S4.	*Mixed duration*: minimise number of exams with different durations that are scheduled into the same room.
S5.	*Larger examinations schedule late in the timetable*: minimise the number of large exams appear '*late*' of the timetable.
S6.	*Period penalty*: minimise the number of exams scheduled in period with penalty.
S7.	*Room penalty*: minimise the number of exams scheduled in room with penalty.

The quality of the timetable produce is calculated through summation of the soft constraint multiply with the related weight. The formulations are as follows:

$$\min \sum (w1 \cdot S1 + w2 \cdot S2 + w3 \cdot S3) + w4 \cdot S4 + ...$$
$$... + w5 \cdot S5 + w6 \cdot S6 + w7 \cdot S7$$

The penalty weightage differ between the datasets. The weightage of each soft constraint for every datasets is summarised in table 8.2.

Table 8.2 The weight of ITC2007 examination datasets

Datasets	*w1*	*w2*	*w3*	*w4*	*w5*	*w6*	*w7*
Exam-1	5	7	5	10	100	30	5
Exam-2	5	15	1	25	250	30	5
Exam-3	10	15	4	20	200	20	10
Exam-4	5	9	2	10	50	10	5
Exam-5	15	40	5	0	250	30	10
Exam-6	5	20	20	25	25	30	15
Exam-7	5	25	10	15	250	30	10
Exam-8	0	150	15	25	250	30	5

The details of the examination competition track can be found in McCollum *et al.* (2007). Researchers which have investigated this dataset include Muller (2008), Cogos *et al.* (2008), Atusta et al. (2007), De Smet (2008) and Pillay (2008) which was the competition entrants follow with McCollum *et al.* (2009) and Turabieh and Abdullah (2012) that reported their finding after the competition. Muller (2008) won the competition by producing the best result during that time. In 2009 McCollum et al, able to show that the ITC2007 result is solvable and able to produce better result than Muller (2008). Turabieh and Abdullah (2012) manage to outperform some of Muller (2008) results with their hybrid methods. A summary of other researcher results is presented in Table 8.3.

Table 8.3 Summary of other researchers result

Datasets	Muller (2008)	Cogos et al. (2008)	Atsuta et al. (2008)	De Smet (2008)	Pillay (2008)	Mc Collum et al. (2009)	Turabieh and Abdullah (2012)
Exam-1	4,370	5,905	8,006	6,670	12,035	4,633	**4,368**
Exam-2	400	1,008	3,470	623	3,074	405	**390**
Exam-3	10,049	13,862	18,622	–	15,917	**9,064**	9,830
Exam-4	18,141	18,674	22,559	–	23,582	**15,663**	17,251
Exam-5	**2,988**	4,139	4,714	3,847	6,860	3,042	3,022
Exam-6	26,950	27,640	29,155	27,815	32,250	**25,880**	25,995
Exam-7	4,213	6,683	10,473	5,420	17,666	**4,037**	4,067
Exam-8	7,861	10,521	14,317	–	16,184	**7,461**	7,519

8.2 Experimental setup

In this work, we implemented the graph heuristics (chapter 4), *modified*-GDA (chapter 6) and *multi-neighbourhood* GDA (chapter 7) using the suggested variable reported in previous chapters. In the graph heuristics, we implemented candidates list 1 and 5. In *modified*-GDA, the Nh1 and Nh7 is used as both able to produce a good quality solution in chapter 5 (compared to other neighbourhood) and finally, the *multi-neighbourhood* GDA technique. In the improvement phase, we use the best found solution in graph heuristics as the initial value. Table 8.4, table 8.5 and table 8.6 show the results of the techiques mention above. Each experiment was run 10 times on a Pentium core2 processor. In *modified*-GDA and *multi-neighbourhood* GDA, we ran for 2000 and 5000 iterations.

8.3 Examination assignment: Results

In this section, we show the result produce using graph heuristics with candidates list, *modified*-GDA and *multi-neighbourhood* GDA. Comparing the result with other researcher from table 8.3 and our result in table 8.4, our graph heuristics (with $c1$ and $c5$) unable to produce a competetive result for all of the exam datasets except for Exam-1, Exam-6 and Exam-8 using $c5$ on Pillay (2008). We extend the search using candidates list 20 ($c20$) and the technique able to produce competetive result compared to the result in table 8.3. Eventhough it unable to outperform the best reported results (as in table 8.3), with $c20$ the result produce able to outperform Atsuta et al. (2008) and De Smet (2008). However, note that, $c20$ takes more computational time around 10,000 seconds and this depending on the number of exams and resources (i.e timeslot and rooms).

Table 8.4 The best found result using graph heuristics

Dataset	*c*1	*c*5	*c*20
Exam-1	27197	9025	6710
Exam-2	31820	3093	949
Exam-3	85577	26765	17102
Exam-4	-	-	-
Exam-5	111724	19160	6643
Exam-6	46165	30775	28255
Exam-7	65165	10203	6582
Exam-8	87340	14473	6539

In *modified*-GDA, we used the initial value from table 8.4. For example, in Exam-1, the initial value is 27197, 9025 and 6710 that correspond to *c*1, *c*5 and *c*20. In Exam-1 with 2000 and 5000 iterations using *c*1 as the initial value, Nh1 able to produce 69% and 73% of improvement respectively. Nh7 with 2000 and 5000 iterations, the *modified*-GDA able to produce 48% and 50% of improvement respectively. While using *c*5 as the initial value with 2000 and 5000 iterations, Nh1 produces 21% and 27% of improvement respectively, and using Nh7 with 2000 and 5000 iterations produce 24% and 28% of improvement. Finally, with *c*20, the *modified*-GDA using Nh1 with 2000 and 5000 iteration produce 3% and 8% of improvement respectively. Nh7 with 2000 and 5000 iterations, the method produce 10% and 13% of improvement respectively. The rest of the result (with the percentage of improvement) is shown in table 8.5. Generally, referring to table 8.3 and table 8.5, the result shows that the *modified*-GDA unable to outperform the best reported result (i.e. Muller, 2008) but able to compete with Atsuta et al. (2008), De Smet (2008) and Pillay (2008). The experiments also reveal that Nh1 able to produce better improvement value compared to Nh7 for all the datasets and a larger number of iterations able to give better improvement value.

In *multi-neighbourhood* GDA, generally the results (see table 8.6) produce is better than *modified*-GDA. However, the approach unable to outperform the best reported result (i.e. Muller, 2008, see table 8.3), but it is able to compete with the Cogos *et al.* (2008) result. In Exam-1, using *c*1 as the initial value, the method able to produce 71%

Table 8.5 The best found results using *Modified*-GDA

Dataset	*c1*				*c5*				*c20*			
	Nh1	*Nh7*	*Nh1*	*Nh7*	*Nh1*	*Nh7*	*Nh1*	*Nh7*	*Nh1*	*Nh7*	*Nh1*	*Nh7*
	2000 iterations		*5000 iterations*		*2000 iterations*		*5000 iterations*		*2000 iterations*		*5000 iterations*	
Exam-1	8296 (69%)	14028 (48%)	7358 (73%)	13660 (50%)	7103 (21%)	6863 (24%)	6598 (27%)	6516 (28%)	6478 (3%)	6072 (10%)	6178 (8%)	5856 (13%)
Exam-2	873 (97%)	6909 (78%)	698 (98%)	6217 (80%)	718 (77%)	1648 (47%)	639 (79%)	1517 (51%)	574 (39%)	804 (15%)	557 (41%)	793 (16%)
Exam-3	20622 (76%)	39227 (54%)	18424 (78%)	33510 (61%)	15117 (44%)	20696 (23%)	14206 (47%)	19753 (26%)	13179 (23%)	16551 (3%)	12489 (27%)	16454 (4%)
Exam-4	-	-	-	-	-	-	-	-	-	-	-	-
Exam-5	5885 (95%)	40005 (64%)	5098 (95%)	39703 (64%)	5558 (71%)	11093 (42%)	4828 (75%)	10801 (44%)	4377 (34%)	4952 (25%)	4183 (37%)	4828 (27%)
Exam-6	34745 (25%)	40275 (13%)	34440 (25%)	40125 (13%)	30765 (0.03%)	30770 (0.02%)	30690 (0.3%)	30745 (0.1%)	27480 (3%)	27625 (2%)	27475 (3%)	27605 (2%)
Exam-7	7240 (89%)	38435 (41%)	6658 (90%)	37373 (43%)	6064 (41%)	6635 (35%)	5516 (46%)	6236 (39%)	5352 (18%)	5793 (11%)	4984 (24%)	5482 (16%)
Exam-8	9867 (88%)	14812 (83%)	9828 (89%)	13945 (84%)	10116 (30%)	10710 (26%)	9786 (32%)	10359 (28%)	9656 (11%)	9825 (9%)	9155 (16%)	9676 (11%)

(*x*%): *show the percentage of improvement compared to the initial solution*

and 74% of improvement with 2000 and 5000 iterations respectively. While using $c5$ as the initial value, it produces 26% and 29% of improvement with 2000 and 5000 iterations respectively. Finally, using $c20$, the technique produce 8% and 12% of improvement with 2000 and 5000 iterations respectively. The rest of the result is shown in table 8.6. The experiment shows that the use of *multi-neighbourhood* GDA able to produce better results than *modified*-GDA. Additionally, using larger number of iterations able to give better improvement value.

Table 8.6 The best found results using *Multi-neighbourhood* GDA

Dataset	*c1*		*c5*		*c20*	
	2000 iterations	*5000 iterations*	*2000 iterations*	*5000 iterations*	*2000 iterations*	*5000 iterations*
Exam-1	7977 (71%)	7052 (74%)	6674 (26%)	6401 (29%)	6169 (8%)	5918 (12%)
Exam-2	1510 (95%)	738 (98%)	756 (76%)	652 (79%)	581 (39%)	533 (44%)
Exam-3	20987 (75%)	18389 (79%)	15458 (42%)	14284 (47%)	13086 (23%)	12589 (26%)
Exam-4	-	-	-	-	-	-
Exam-5	8209 (93%)	5224 (95%)	6462 (66%)	4849 (75%)	4283 (36%)	4064 (39%)
Exam-6	37410 (19%)	37035 (20%)	30685 (0.3%)	29220 (5%)	27409 (3%)	27480 (3%)
Exam-7	8922 (86%)	6807 (90%)	6055 (41%)	5467 (46%)	5334 (18%)	5081 (22%)
Exam-8	10350 (88%)	9560 (89%)	10117 (30%)	9674 (33%)	9604 (11%)	9181 (15%)

(x%): *show the percentage of improvement compared to the initial solution*

8.4 Discussion

In graph heuristics the technique able to produce a competitive result compared with other researcher results only when using high value of candidate list but suffer an increase of computational times. For example, in Exam-1 using $c20$, the run takes around 10,000 seconds and this dependent on the number of exams, rooms and timeslots. An increase in the number of these variables would eventually increase the computational times. This is because the candidate list will compare each variable and choose the location that returns less cost values.

In *modified*-GDA, the technique able to produce competitive results comparable to the result in table 8.3. Generally, it produces a high percentage of improvement with $c1$. The result shows that Nh1 able to give a better improvement value compared to Nh7. This support our finding in chapter 6 that Nh1 is superior compares to other neighbourhoods in the experiments. In *multi-neighbourhood* GDA, the technique able to produce competitive result compared to result in table 8.3. Furthermore, it is able to produce better results than *modified*-GDA and this supports our finding in chapter 7 where *multi-neighbourhood* GDA able to produce better results than *modified*-GDA.

In our observation, we notice that the result produce by the improvement phase dependent on the initial solution. Hence, having a good initial solution help to speeds the search for a better solution. Furthermore, having a large number of iteration help the algorithm to explore search space. Finally, based on the result shown in table 8.4, 8.5 and 8.6 we can classify the dataset based on the following category.

a) **Time consuming exam:** Exam-2, Exam-3 and Exam-7 are the most time consuming datasets. This is because of the large number of exams to schedule as well as a large number of timeslots and rooms to choose from which increases the search time. Even though Exam-5 have a large number of exams (i.e. 1018), it contains a small number of rooms to choose from.
b) **Challenging exam:** Exam-6 and Exam-4 the most challenging exam as we are even struggling to produce a feasible solution. Both exams have a high conflict density, additionally, in Exam-6 it has a large number of exams involve in COINCIDENCE constraints. As Exam-4, it has the high number of exam involve in the EXCLUSION constraint. This constraint alone forced the exam (EXCLUSION) to be scheduled to four different timeslot. For Exam-4, we were unable to produce a feasible solution.
c) **Highly constraints:** in our opinion Exam-5 and Exam-8 are the most highly constraints dataset. This is because of the exams that involve in multi period hard constraints (e.g. *examA* AFTER *examB* and *examA* COINCIDENCE *examC*)

Even though with a different level of complexity on each dataset, our proposed method able to works in producing a feasible solution and competitive results.

8.5 Contributions

This work has presented a study of a real-world examination timetabling problem from the ITC2007 competition examination track. The problem involves scheduling exams into timeslots and rooms for eight datasets that have different constraints. The contributions of this work are as follows:

a) We have implemented the graph heuristic, *modified*-GDA and *multi-neighbourhood* GDA to the ITC2007 datasets. These methods able to solve the ITC2007 except for Exam-4.
b) We have shown that the proposed methods able to produce a competitive result compared with other works reported in the literature.
c) We have classifies the datasets into three main categories that is time consuming, challenging and highly constraint datasets. This information could aid in understanding the dataset in order to produce a better result.

8.6 Conclusion

In this chapter, we have investigated a real world examination timetabling problem, ITC2007 using graph heuristics, *modified*-GDA and *multi-neighbourhood* GDA. We can conclude that the proposed method able to produce a competitive solution compared to other reported works. Even though the proposed method unable to outperform the best reported result (i.e. Muller, 2008) but the experiment support our claim from the previous chapters that include:

a) In a single neighbourhood (i.e. *modified*-GDA), Nh1 proof able to give better improvement value because of it explorative nature.
b) In multi-neighbourhood strategy (i.e. *multi-neighbourhood* GDA), it increases the chance of producing better results than single neighbourhood.
c) A larger number of iterations increases the chances of producing better solutions.

Chapter 9

Conclusion and Future Research Directions

This chapter summaries the work reported in this thesis. Section 9.1 gives a summary of the research that has been carried out. The scientific contributions are described in Section 9.2. Section 9.3 and 9.4 outlines further research directions that may be undertaken and final reflections of the research.

9.1 Research work summary

The investigated research is concerned with a real world examination timetabling problem taken from the Universiti Malaysia Pahang (UMP). The UMP examination timetabling process involves assigning exams to timeslots and rooms, and scheduling invigilators. The investigated examination dataset contains additional constraints, when compared to others constraints reported in the scientific literature. A comparison of the constraints is presented in chapters 2 and 3. Additionally, we construct an invigilator schedule, which has largely been ignored in the scientific community.

The UMP examination timetabling problem is solved in two phases, firstly scheduling the exams into timeslots and rooms (*exam-timeslot-room* assignment), and secondly scheduling the invigilators based on phase one. In solving the *exam-timeslot-room* assignment, we present a formal model of the UMP problem in chapter 4. We have

implemented graph heuristics with candidates lists in constructing an initial solution. This work has been published in the European Journal of Operational Research, EJOR (Kahar and Kendall, 2010a). Next, we present the formal model of the UMP invigilator scheduling problem in chapter 5. The work is currently under review for the Journal of Operational Research Society, JORS. We have also include additional constraints (in addition to the original UMP invigilation constraints) considering the comments reported in a survey by Awang et al. (2006). Based on these experiments (on invigilators scheduling), the results reveal that the invigilator scheduling result is dependent on the number of rooms being selected from the *exam-timeslot-room* assignment phase. Henceforth, we concentrated only on improving the initial result of the *exam-timeslot-room* assignment.

An improvement methodology involves *modified*-GDA and *multi-neighbourhood* GDA approaches. The new method is designed with the timetable officer in mind as it uses a simple to understand parameter for ease of operation. The *modified*-GDA approach is described in chapter 6. The *modified*-GDA uses a simple to determine parameter and is capable of adjusting the desired value, boundary and decay rate to guide to search for better solution than the initial cost (while using good neighbourhood heuristics). A statistical analysis, reveals that the choice of neighbourhood heuristics, number of iterations and the initial solution plays a significant role in producing a good quality solution. This work is currently under review for the Journal of Operational Research Society, JORS. The results presented in chapter 6, shows that the choice of the neighbourhood heuristics is very important.

We then extend the *modified*-GDA by presenting a *multi-neighbourhood* GDA approach. This work is presented in chapter 7. The method uses more than one neighbourhood in order to effectively explore the search space and improve the solution. Furthermore, the multi-neighbourhood simplifies the operation of the algorithm for the timetable officer by not having to determine the suitable neighbourhoods. The multi-neighbourhood approach is able to generate better quality solutions when compared to *modified*-GDA.

Finally, we investigate the examination track of the Second International Timetabling Competition (ITC2007) using the proposed methodology mention above. The proposed method is able to produce competetive result when compared to other work in the scientific literature.

9.2 Contributions

The overall research contributions can be categorised into contributions to the scientific community and contributions to the institution (UMP). They are identified below.

1) **Develop a formal model of the UMP examination timetabling problem:**

Contribution to the scientific community: We develop a formal model of the UMP *exam-timeslot-room* timetabling problem (see chapter 4) and the UMP *invigilator* scheduling problem including additional invigilator constraints from Awang et al. (2006) (see chapter 5). The *exam-timeslot-room* timetabling problem contains new constraints which are different to other datasets presented in the scientific literature.

Contribution to the institution (UMP): We have documented the *exam-timeslot-room* and *invigilator* timetable requirements (constraints) which have never been documented before in UMP. Furthermore, the formal model will be useful for future assesment of the UMP examination timetable solution. Additionally, we also consider extra constraints for the invigilator scheduling based on the invigilator comments (Awang et al. 2006) which we believe closely reflect the UMP invigilator scheduling needs.

2) **Construction of initial solution:**

Contribution to the scientific community: We have utilised graph heuristics that call upon candidate lists for the UMP examination timetabling problem. The

approach is able to produce good quality solutions within reasonable computational times, when compared to the UMP proprietary software. Some of the interesting aspects of the work we report include:

- Candidate list feature that choose multiple resources (timeslot and room) and selects the resources that contribute to lower penalties value. This allows the algorithm to find a good initial solution, which is then used in the improvement phase.
- The pre-determined room grouping allow for fast room(s) selection. It also allows for minimising the spreading (F_2) and splitting (F_1) cost penalty.

We also implemented the same approach to the ITC2007 dataset and it able to produce competitive results when compared to other work reported in the scientific literature.

Contribution to the institution (UMP): Development of UMP examination timetabling system, which includes assigning exams to timeslots and rooms, and scheduling invigilators. The timetable produced complies with the constraints which the UMP proprietary system fails to achieve.

3) **Improving the initial solution:**

Contribution to the scientific community: We have proposed a modified great deluge algorithm (*modified*-GDA) to improve on the constructive heuristic solutions for the UMP exam problem. The *modified*-GDA uses a single parameter which benefits the timetable officer in operating the systems. The methodology is able to produce good quality solutions when applied to the UMP examination datasets. Additionally, we investigate great the deluge algorithm parameter settings which includes different initial solutions, different number of iterations and different neighbourhood heuristics for the *modified*-GDA. Statistical analysis determines whether there are significant differences between different parameters settings. The investigation revealed that the choice of parameter plays an important role in the search.

We applied the modified great deluge algorithm with multi-neighbourhood heuristics (*multi-neighbourhood* GDA) to the UMP exam problem. The *multi-neighbourhood* GDA able to generate better quality solution when compared to the original *modified*-GDA for the UMP examination problem. Additionally, we investigated the neighbourhood heuristics (by removing some of the neighbourhoods), revealing that it is best to use the best neighbourhood during the search and is also worthwhile using the entire neighbourhood to encourage exploration. The multi-neighbourhood simplifies the operation of the algorithm for the timetabling officer, rather than having to determine the suitable set of neighbourhoods. This is beneficial as in Kahar and Kendall (2010b), we show that the choice of neighbourhood plays a major role in the search.

Contribution to the institution (UMP): Implementation of a *modified*-GDA and *multi-neighbourhood* GDA approach that uses a simple parameter to allow easy operation by the timetable officer.

4) **Implementation to ITC2007 datasets**

The graph heuristics with candidate lists, *modified*-GDA and *multi-neighbourhood* GDA were implemented for the ITC2007 examination datasets. This is to ascertain that the proposed methodology is able to work with another exam timetabling problem. We are able to generate competitive results compare to other results reported in the scientific literature.

9.3 Future research directions

It is recognised that a gap exists between theory and practice in examination timetabling. Different institutions have different requirements (constraints) and it is difficult to produce a common solution methodology. This thesis has focused on solving a real world examination timetabling problem that includes scheduling exams to timeslots and rooms as well as scheduling invigilators. We also investigate several new methodologies for solving the problem. The results achieved are better than the proprietary software currently used. The proposed methodologies are also effective on

the ITC2007 datasets. However, there are several future research directions that we identify below.

9.3.1 Improving the proposed approach

As seen in the previous chapters, this work has concentrated on attempting to solve the UMP examination timetabling problem using graph heuristics, *modified*-GDA and *multi-neighbourhood* GDA. The graph heuristics (chapter 4) are able to produce good quality solutions, normally using a high candidate list value. However as the candidates list size increased, the algorithm takes a considerable more computational time. Even so, in real world situations the time to produce the examination timetable is not usually time critical (within sensible limits). This is due to the fact that the process of generating the examination timetable is normally carried out two to three month before the exams take place. However, it is worth investigating ways of reducing the running time (especially for the ITC2007 datasets). This could be done by including a look-ahead mechanism that lists the available timeslot for the next exams to scheduled. Additionally, having information of the spreads between scheduled exams and the next to be scheduled exam would reduce the time of selecting timeslot with minimum spreading penalty value.

The *modified*-GDA and *multi-neighbourhood* GDA approach, in the improvement phase, allows the boundary, that acts as the acceptance level to dynamically change during the search. Currently the boundary is set to change based on a constants value. A further exploration can be done by implementing a dynamic based value during the boundary tuning. This might include a dynamic desired value and dynamic boundary value. These values could aid in exploring the search space. Morevover, a further investigation on the *multi-neighbourhood* GDA could include combining different neighbourhood heuristics. We believe that this could save computational time if suitable neighbourhoods are combined in an intelligent ways

9.3.2 Hybridisation

The proposed approaches are open for hybridisation with other methods. For example, the hybridisation of graph heuristics with candidate lists together with fuzzy logic. Fuzzy logic could be used to select the (next) exams to be scheduled instead of pre-arranging them using the graph colouring method (i.e. Largest enrollment, largest degree, etc). Additionally, future investigation on the graph heuristics with candidate lists could involve hybridisation with meta-heuristic methods (e.g. hill climbing, great deluge algorithm, etc). Possibilities include ways to partially schedule the exams using graph heuristic (based on a pre-determined constrant) and using meta-heuristics to improve the partially schedule exam based on a pre-determined number of improvement cycles. Furthermore, hybridisation of the *modified*-GDA and *multi-neighbourhood* GDA with tabu search could also be investigated. The tabu search could be used to hold visited points in the search space and thus avoid cycling. Alternatively, hold the unperforming neighbourhood heuristics in the multi-neighbourhood approach.

9.3.3 Invigilator scheduling

In this research, we have developed a formal model for the invigilator scheduling as well as included additional constraints in addition to the UMP original constraints. Some further investigation could include investigating the optimal number of invigilators required for an examination timetable which could help to minimise the operational cost instead of selecting non-academic staff (as this takes them away from other duties). It might also be worthwhile investigating automated system that is able to assist in determining the effect of constraints on the objective value so that the effect of performing swaps between the invigilators can be evaluated. Additionally, it could provide a suggestion (or list of availabilities) in making moves or swapping the invigilation duties.

9.3.4 Dynamic timetabling system

Based on the discussion with the timetable officer, they often receive last minute requests for changes to the timetable. This sometimes includes last minute examination paper additions. Hence, the timetable officer could re-run the whole examination timetable or simply insert the (late) requested exam into the current (complete) timetable, aiming for minimal disruption. In the latter approach, it is worth investigating ways of satisfying all hard constraints and minimising the penalty value, with minimal disruptions to already schedule exams.

9.4 Final reflections

In this research, we bridge the gap between research and practice by investigating a problem taken from Universiti Malaysia Pahang (UMP) that has several novel constraints, in addition to those commonly used in the scientific literature. We have implemented graph heuristics with candidate list, *modified*-GDA and *multi-neighbourhood* GDA to solve the UMP examination timetabling problem. These methods show able to produce better results than the proprietary software currently used. With this, UMP now has access to a set of high-quality algorithms that were not available before this research was undertaken. Moreover, the algorithm been shown to be effective on other problems, particularly the ITC datasets. As such, the timetabling community is able to benefit from the approaches presented in this work. We hope that this work will motivate other researchers to further improve on the methodologies presented in this thesis.

Bibliography

1. Abdullah S, (2006). Heuristic Approaches for University Timetabling Problems. Ph.D. Thesis, School of Computer Science and Information Technology, University of Nottingham, June 2006.
2. Abdullah S, Ahmadi S, Burke E K, Dror M and McCollum B, (2007). A tabu based large neighbourhood search methodology for the capacitated examination timetabling problem, Journal of Operational Research Society, 58(11), pages 1494-1502.
3. Abdullah S, Burke E K and McCollum B, (2005). An investigation of variable neighbourhood search for university course timetabling. In Proceedings of MISTA 2005: The 2nd Multidisciplinary Conference on Scheduling: Theory and Applications, pages 413-427, NY, USA, 18-21 July 2005.
4. Abdullah S, Shaker K, McCollum B, and McMullan P, (2009). Construction of Course Timetables Based on Great Deluge and Tabu Search, Proceedings of MIC 2009, VIII Meteheuristic International Conference, Hamburg, July 13-16.
5. Abdullah S, Turabieh H and McCollum B, (2009). A Tabu-based Memetic Approach for Examination Timetabling Problems. AI-2009 Twenty-ninth SGAI International Conference on Artificial Intelligence., December 15-17, Cambridge, England.
6. Abdullah S, Turabieh H, (2012). On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems, Information Sciences, Vol. 191, Pages 146-168.
7. Acan A and Tekol Y, (2003). Chromosome reuse in genetic algorithms. In Proceedings of the 2003 genetic and evolutionary computation conference (GECCO 2003), pages 695–705).
8. Ahandani M A, Baghmisheh M T V, Zadeh M A B, Ghaemi S, (2012). Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem, Swarm and Evolutionary Computation, Available online 9 July 2012, ISSN 2210-6502, 10.1016/j.swevo.2012.06.004.
9. Ahuja R K, Orlin J B and Sharma D D, (2000). Very large scale neighbourhood search. International Transactions in Operational Research, 7, pages 301-317.
10. Al-Betar M A and Khader A T, (2008). A harmony search algorithm for university course timetabling. Annals OR 194(1): 3-31

11. Asmuni H, Burke E K, Garibaldi J and McCollum B, (2004). 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.

12. Asmuni H, Burke E K, Garibaldi J M and McCollum B, (2005). Fuzzy Multiple Heuristic Orderings for Examination Timetabling in Selected Papers from Practice and Theory of Automated Timetabling V, Volume 3616, pages 334-353. Springer, 2005.

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

14. Awang S, Kahar M N M, Jamil N M and Ajid K A, (2006). A Satisfaction Survey on KUKTEM Automated Examination Scheduling. Malaysian Science and Technology Congress 2006 (MSTC 2006).

15. Ayob M, Abdullah S and Malik A M A, (2007). A Practical Examination Timetabling Problem at the Universiti Kebangsaan Malaysia, International Journal of Computer Science and Network Security , 7(9), pp 198-204.

16. Ayob M, Hamdan A R, Abdullah S, Othman Z, Nazri M Z A, Razak K A, Tan R, Baharom N, Ghafar H A, Dali R M, Sabar N R, (2011). Intelligent Examination Timetabling Software, Procedia - Social and Behavioral Sciences, Volume 18, 2011, Pages 600-608.

17. Ayob M, Malik A M A, Abdullah S, Hamdan A R, Kendall G and Qu R, (2007). Solving a Practical Examination Timetabling Problem:A Case Study. In O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, Lecture Notes In Computer Science (LNCS) 4707, Part III, Springer-Verlag Berlin Heidelberg, pp. 611–624.

18. Azimi Z N, (2005). Hybrid heuristics for Examination Timetabling problem, Applied Mathematics and Computation, Volume 163, Issue 2, 15 April 2005, Pages 705-733,

19. Azimi, Z N, (2004). Comparison of metaheuristic algorithms for examination timetabling problem. Journal of Applied Mathematics and Computing, 16 (1-2), pp. 337-354.

20. Balakrishnan N, (1991). Examination scheduling: A computerized application, *Omega*, 19, issue 1, pages. 37-41,

21. Balakrishnan N, Lucena A and Wong R T, (1992). Scheduling examinations to reduce second order conflicts. Computers and Operations Research, 19, pages 353-361.

22. Bardadym V A, (1996). Computer-aided school and university timetabling: a new wave. The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), pp 22-45.

23. Barham A M and westwood J B, (1978). A simple heuristic to facilitate course timetabling. J. Opl res. Soc. 29, pages 1055-1060.

24. Brelaz D, (1979). New methods to colour the vertices of a graph. Communication of ACM, 22, pages 251-156.

25. Broder S, (1964). Final Examination Scheduling, Communications of the ACM 7 (1964), pages 494-498

26. Bullnheimer B, (1998). An examination scheduling model to maximize students' study time. E. Burke, m. Carter, eds. The practice and theory of automated timetabling: selected papers (Patat '97). Lecture notes in computer science 1408. Springer-Verlag, Berlin Heidelberg, New York, pages 78-91.

27. Burke E K and Bykov Y, (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. In: Burke, E., Gendreau, M., (eds.): Proc. of the 7th International Conference on the Practice and Theory of Automated Timetabling, Montreal (2008)

28. Burke E K and Carter M W (Eds) (1998). Practice and Theory of Automated Timetabling II, 2nd International Conference, PATAT'97, Toronto, Canada, August 20-22, 1997, Selected Papers. Lecture Notes in Computer Science. vol. 1408 Springer 1998, ISBN 3-540-64979-4.

29. Burke E K and De Causmaecker P (eds) (2003). Practice and Theory of Automated Timetabling IV, 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002, Selected Revised Papers. Lecture Notes in Computer Science 2740 Springer 2003, ISBN 3-540-40699-9.

30. Burke E K and Erben W, (Eds) (2001). Third International Conference on Practice and Theory of Automated Timetabling III, PATAT 2000, Konstanz, Germany, August 16–18, 2000, Selected Papers. Lecture Notes in Computer Science, 2079. Springer, ISBN 3-540-42421-0.

31. Burke E K and Kendall G, (Eds) (2005). Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, Springer 2005.

32. Burke E K and Landa-Silva J, (2004). The Design of Memetic Algorithms for Scheduling and Timetabling Problems. In E. Willaim, N. Krasnogor, and J. Smith,

editors, Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, volume 166, pages 289-312. Springer, 2004.

33. Burke E K and Newall J (2004). Solving examination timetabling problems through adaptation of heuristic orderings. Annals of operations Research, 129:107-134.

34. Burke E K and Newall J P (2006). Solving examination timetabling problems through adaption of heuristic ordering, Annals of Operations Research, 129(2), pages 107-134.

35. Burke E K and Newall J P, (2003). Enhancing timetable solutions with local search methods. Practice and Theory of Automated Timetabling IV: Selected Papers from the 4th International Conference. Springer Lecture Notes in Computer Science, vol. 2740. pages 195-206.

36. Burke E K and Newall J, (1999). A multistage evolutionary algorithm for the timetable problem, Evolutionary Computation, IEEE Transactions on , vol.3, no.1, pp.63-74, Apr 1999.

37. Burke E K and Newall J, (2002). Enhancing timetable solutions with local search methods. Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference. Springer Lecture Notes in Computer Science, vol. 2740, pages 195-206.

38. Burke E K and Petrovic S, (2002). Recent research directions in automated timetabling. European Journal of Operational Research 140 (2), pages 266–280.

39. Burke E K and Ross P (Eds) (1996). Practice and Theory of Automated Timetabling, 1st International Conference, Edinburgh, U.K., August 29 - September 1, 1995, Selected Papers. Lecture Notes in Computer Science, vol. 1153 Springer 1996, ISBN 3-540-61794-9.

40. Burke E K and Rudova H. (eds) (2007). Practice and Theory of Automated Timetabling VI, 6th International Conference, PATAT 2006, Brno, Czech Republic, August 30 - September 1, 2006, Revised Selected Papers. Lecture Notes in Computer Science 3867 Springer 2007, ISBN 978-3-540-77344-3.

41. Burke E K and Trick M (Eds) (2005). Practice and Theory of Automated Timetabling V, 5th International Conference, PATAT 2004, Pittsburgh, PA, USA, August 18-20, 2004, Revised Selected Papers. Lecture Notes in Computer Science 3616 Springer 2005, ISBN 3-540-30705-2.

42. Burke E K, Bykov Y and Petrovic S, (2001). A multicriteria approach to examination timetabling. In: E.K. Burke andW. Erben (eds). (2001). Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Springer Lecture Notes in Computer Science, vol. 2079, pages 118-131.

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

44. Burke E K, Eckersley A J, McCollum B, Petrovic S and Qu R, (2010a). Hybrid variable neighbourhood approaches to university exam timetabling, European Journal of Operational Research, Volume 206, Issue 1, 1 October 2010, Pages 46-53.

45. Burke E K, Eckersley A, McCollum B, Petrovic S and Qu R, (2003). Using Simulated Annealing to Study Behavior of Various Exam Timetabling Data Sets, 5th Meta-heuristics International Conference MIC03, August 25 - 28 2003, Kyoto International Conference Hall, Kyoto, Japan.

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

47. Burke E K, Elliman D G and Weare R F, (1995). The Automation of the Timetabling Process in Higher Education, Journal of Educational Technology Systems, 23(4): 257-266.

48. Burke E K, Elliman D G, Ford P H and Weare R F (1996). Examination timetabling in British universities - A survey. In Burke, E.K. and Ross, P. (eds) (1996). The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling, Edinburgh, UK, Lecture Notes in Computer Science, vol. 1153, Springer-Verlag, pages 76-92.

49. Burke E K, Hyde M, Kendall G, Ochoa G, Ozcan E and Qu R, (2010b). Hyper-heuristics: A Survey of the State of the Art, School of Computer Science and Information Technology, University of Nottingham, Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747.

50. Burke E K, Hyde M, Kendall G, Ochoa G, Özcan E, and Woodward J. (2009). A Classification of Hyper-heuristics Approaches, School of Computer Science and Information Technology, University of Nottingham, Computer Science Technical Report No. NOTTCS-TR-SUB-0907061259-5808.

51. Burke E K, Kendall G and Soubeiga E, (2003). A Tabu-Search Hyper-Heuristic for Timetabling and Rostering. Journal of Heuristics, 9(6), pages 451-470.

52. Burke E K, Kingston J H and de Werra D, (2004). Applications to timetabling. In: J. Gross and J. Yellen (eds.) The Handbook of Graph Theory, Chapman Hall/CRC Press, 2004, pages 445-474.

53. Burke E K, Landa-Silva D and Soubeiga E, (2005). Multi-objective hyperheuristic approaches for space allocation and timetabling, in T. Ibaraki, K. Nonobe, M. Yagiura (eds.) Meta-heuristics: Progress as Real Problem Solvers, selected papers from the 5th Metaheuristics International Conference (MIC 2003), Springer, pages 129-158.

54. Burke E K, McCollum B, McMullan P and Qu R, (2006). Examination timetabling: A new formulation. Extended Abstract. Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling. 30th August-1st September 2006, Brno, The Czech Republic, pp 373-375.

55. Burke E K, McCollum B, Meisels A, Petrovic S, and Qu R, (2007). A Graph-Based Hyper-Heuristic for Educational Timetabling Problems European Journal of Operational Research, 176:177-192.

56. Burke E K, Newall J and Weare R F, (1996). A Memetic algorithm for university exam timetabling. In: Burke, E.K., Ross, P. (Eds.), Selected Papers from the First International Conference on Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 1153. Springer-Verlag, pages 241–250.

57. Burke E K, Newall J, and Weare R F, (1998). E.K. Burke, J. Newall, and R. Weare, A simple heuristically guided search for the timetabling problem. Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), pages 574-579.

58. Burke E K, Petrovic S and Qu R, (2006). Case Based Heuristic Selection for Timetabling Problems. Journal of Scheduling, 9: pages 115-132.

59. Caramia M, Dell'Olmo P and Italiano G F (2001). New Algorithms for Examination Timetabling. In: Naher, S., Wagner, D. (eds.): Algorithm Engineering 4th International Workshop, Proceedings WAE 2000. Lecture Notes in Computer Science, vol. 1982. Springer-Verlag, pages 230-241.

60. Carrington J, Pham N, Qu R and Yellen J, (2007). An Enhanced Weighted Graph Model for examination/Course Timetabling In Proceedings of PlanSIG 2007, (Bartak R., Ed.), 26th Workshop of the UK Planning and Scheduling SIG 26, pages 9-16.

61. Carter M W (1986). A survey of practical applications of examination timetabling algoritms. Operations Research, 34, pages 193-202.

62. Carter M W and Gendreau M, (1992). A practical algorithm for finding the largest clique in a graph. Centre for Research on Transportation, vol 820, pages 266 – 272.

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

64. Carter M W and Laporte G (1996). Recent developments in practical examination timetabling. In: Burke, E.K. and Ross, P. (eds) (1996). Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling I, Edinburgh, UK, Lecture Notes in Computer Science, vol. 1153, Springer-Verlag, pages 3-21.

65. Carter M W and Laporte G (1998). Recent developments in practical course timetabling. The Practice and Theory of Automated Timetabling II: Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag. (Editors: E.K. Burke and M. Carter), pages 3-19.

66. Carter M W, (2001). A comprehensive course timetabling and student scheduling system at the University of Waterloo. The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 64-82.

67. Carter M W, Laporte G, Lee S Y (1996). Examination timetabling: Algorithmic strategies and applications. Journal of Operational Research Society 47 (3), 373–383.

68. Chiarandini M, Birattari M, Socha K and Rossi-Doria O, (2006). An effective hybrid algorithm for university course timetabling. Journal of Scheduling, vol. 9, no. 5, pp. 403–432

69. Chu S. C. and Fang H. L. (1999). Genetic algorithms vs. tabu search in timetable scheduling," in Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, 1999, Pages. 492–495.

70. Corne D, Fang H-L and Mellish C, (1993). Solving the Module Exam Scheduling Problem with Genetic Algorithms. Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Chung, Lovegrove and Ali (eds.), 1993, pages 370-373.

71. Costa D and Hertz A, (1997). Ants can colour graphs, Journal of the Operational Research Society 48, pages 295-305.

72. Costa D, (1994). A tabu search algorithm for computing an operational timetable. European Journal of Operational Research, 76: 98-110.

73. Cote P, Wong T and Sabouri R, (2005). Application of a hybrid multi-objective evolutionary algorithm to the uncapacitated exam proximity problem. Practice and Theory of Automated Timetabling: Selected Papers from the 5th International Conference. Lecture Notes in Computer Science, vol. 3616, pp 151-168.

74. Cowling P, Kendall G and Hussin N M, (2002). 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 2002), Gent, Belgium, pp 258-261.

75. Cuupic M, Golub M and Jakobovic D, (2009). Exam timetabling using genetic algorithm. ITI 2009: 357-362.

76. Dammak A, Elloumi A K and Kamoun H, (2006). Classroom assignment for exam timetabling, Advances in Engineering Software. Volume 37, Issue 10, , October 2006, Pages 659-666.

77. Dawkins R, (1976). The Selfish Gene, Oxford University Press, Oxford.

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

79. de Werra D, (1996a). Extensions of colouring models for scheduling purposes. European Journal of Operational Research, 92, pp 474-492.

80. de Werra D, (1997). Theory and methodology: The combinatorics of timetabling. European Journal of Operational Research, 96, pages 504-513.

81. Di Gaspero L and Schaerf A, (2001). Tabu search techniques for examination timetabling. In: Burke, E.K and Erben, W (eds). (2001). Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference. Lecture Notes in Computer Science, vol. 2079. Springer-Verlag, pp 104-117.

82. Di Gaspero L, (2002). Recolour, shake and kick: A recipe for the examination timetabling problem. In: E.K. Burke and P. De Causmaecker (eds) (2002). Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. 21st -23rd August 2002. KaHo St.-Lieven, Gent, Belgium. Pages 404-407.

83. Dimopoulou M and Miliotis P, (2001). Implementation of a university course and examination timetabling system, European Journal of Operational Research Volume 130, Issue 1, 1 April 2001, Pages 202-213.

84. Dorigo M, Maniezzo V and Colorni A, (1996). The ant system: optimization by a colony of cooperating agents, IEEE Trans. on System, Man, and Cybernetics-part B, Vol.26, No. 1, pages 29-42.

85. Dueck G, (1993). New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel, Journal of Computational Physics, Volume 104, Issue 1, January 1993, Pages 86-92,

86. Eley M, (2006). Some experiments with ant colony algorithms for the exam timetabling problem. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4150 LNCS, pages. 492-499.

87. Eley M, (2007). Ant algorithms for the exam timetabling problem. In: E.K. Burke and H. Rudova (eds), Practice and Theory of Automated Timetabling: Selected Papers from the 6th International Conference. Lecture Notes in Computer Science, vol. 3867, pages 364-382.

88. Erben W, (2001). A grouping genetic algorithm for graph colouring and exam timetabling. Lecture Notes in Computer Science vol. 2079. The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000). Burke, E.K. and Erben, W. (eds.), Springer-Verlag, Berlin, Heidelberg, New York, pages 132-156.

89. Frausto S J, Alonso P F, (2008). FRALONSO: A Hybrid Simulated Annealing-Tabu Search Algorithm for Post Enrolment Course Timetabling. In: PATAT Conferences 2008, Montreal Canada.

90. Gen M and Chen R, (1997). Genetic Algorithms and Engineering Design, John Willey & Sons, Inc, New York.

91. Glover F and Laguna M, (1993). Tabu Search. Modern Heuristic Techniques for Combinatorial Problems, C. Reeves (ed.), Blackwell Scientific Publishing, Oxford, pages 70-150.

92. Glover F and Laguna M, (1997). Tabu Search. Norwell, MA: Kluwer Academic Publisher.

93. Glover F, (1986). Future paths for Integer Programming and Links to Artificial Intelligence", Computers and Operations Research, 13: pages 533 - 549.

94. Glover F, (1989). Tabu Search - Part I. ORSA Journal on Computing, 1: 190 - 206.

95. Glover F, (1990). Tabu Search - Part II. ORSA Journal on Computing, 2: 4 - 32.

96. Glover F, Kelly J P and Laguna M (1995). Genetic Algorithms and Tabu Search: Hybrids for Optimization. *Computers and Operations Research.* Vol. 22, No. 1, pages 111 – 134.

97. Gogos C, Alefragis P, and Housos E (2008). A multi staged algorithmic process for the solution of the examination timetabling problem. Proceedings of the 7th International

Conference on Practice and Theory of Automated Timetabling. 19-22 August 2008. University of Montreal, Canada

98. Goldberg D E, (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

99. Hansen P and Mladenovic N, (2005). Variable neighborhood search, in Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, (Burke E. K.and Kendall G, eds.) pp. 211-238, Springer 2005.

100. Hansen P and Mladenovic N. (1999). An Introduction to Variable Neighborhood Search. In S. Voß, S. Martello, C. Roucairol, and I.H. Osman (eds.), Meta-Heuristics 98: Theory & Applications. Norwell, MA: Kluwer Academic Publishers, pp. 433-458.

101. Hertz A, (1991). Tabu search for large scale timetabling problems. Eur. J. Opl Res. 54, pages 39-47

102. Hertz A, Taillard E and de Werra D, (1995). A Tutorial on Tabu Search, Proc. of Giornate di Lavoro AIRO'95, Entreprise Systems: Management of Technological and Organizational Changes, pages 13-24.

103. Holland J. (1975). Adaptation In Natural and Artificial Systems. The University of Michigan Press, Ann Arbour.

104. Hussin N M, (2005). Tabu search based hyper-heuristic approaches to examination timetabling. Phd thesis, University of Nottingham, UK.

105. Jat N S and Yang S (2008). A Memetic Algorithm for the University Course Timetabling Problem, The proceeding of 20th IEEE International Conference on Tools with Artificial Intelligence, vol, pages 1082-3409/08.

106. Johnson D, (1990). Timetabling university examinations. J. Opl Res. Soc. 41, pages 39-47.

107. Kahar M N M and Kendall G, (2010a). The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. European Journal of Operational Research, Volume 207, Issue 2, 1 December 2010, pp 557-565

108. Kahar M N M and Kendall G, (2010b). Universiti Malaysia Pahang Examination Timetabling Problem: Scheduling Invigilators. Under 2nd review for the Journal of the Operational Research Society, JORS

109. Kahar M N M and Kendall G, (2011). A Great Deluge Algorithm for a Real World Examination Timetabling Problem. Under review for the Journal of the Operational Research Society, JORS

110. Kendall G and Hussin M N. (2005b). Tabu search hyper-heuristic approach to the examination timetabling problem at university technology mara. In: Burke, E.K., Trick, M., (Eds.), Proceedings of the Fifth International Conference on the Practice and Theory of Automated Timetabling (PATAT), Pittsburgh, USA, 18–20 August 2004, pp. 199–217.

111. Kendall G and Hussin N M, (2004). A Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology. Selected Papers of the 5th International Conference of Practice and Theory of Automated Timetabling V (PATAT 2004), Burke E. and Trick T. (eds), LNCS 3616, pages 270-293.

112. Kendall G and Hussin N M, (2005). An Investigation of a Tabu Search Based on Hyper-Heuristics for Examination Timetabling, in Kendall G., Burke E.K., Petrovic S. (eds), Proceedings of the 2nd Multidisciplinary Scheduling: Theory and Applications Conference (MISTA 2005), 309–328.

113. Kirkpatrick S, Gelatt Jr C D and Vecchi M P, (1983). Optimization by simulated annealing. Science, 220: 671 - 680.

114. Krasnogor N and Smith J, (2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation, 9(5): 474 – 488.

115. Landa-Silva D and Obit J H, (2008). Great Deluge with Nonlinear Decay Rate for Solving Course Timetabling Problems. Proceedings of the *2008 IEEE Conference on Intelligent Systems (IS 2008)*, IEEE Press, pp. 8.11-8.18.

116. Laporte G and Desroches S (1984). Examination timetabling by computer. Computers and Operations Research. 11, pages 351-360.

117. Leong T Y and Yeong W Y, (1990). A hierarchical decision support system for university examination scheduling. Working Paper, National University of Singapore; 1990. pages 1 - 14..

118. Lim A, Chin A J, Kit H W and Oon W C, (2000). A campus-wide university examination timetabling application. In: Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence, pages 1020 - 1025.

119. Malik A M A and Masri A and Razak H A, (2007). A Heuristic for Scheduling Examination to Room Based on Exam Duration Length. In: ICEEI2007, 17 June 2007, Bandung, Indonesia.

120. Massoodian S and Esteki A, (2008). A hybrid genetic algorithm for curriculum based course timetabling, the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, 18-22 August 2008, Montreal, Canada.

121. McCollum B, (2007). A perspective on bridging the gap between theory and practice in university timetabling. In: Burke, E.K., Rudova, H. (Eds.), Selected Papers from the Sixth International Conference on Practice and Theory of Automated Timetabling. Lecture Notes in Computer Science, vol. 3867, pages 3–23.

122. McCollum B, McMullan P J, Parkes A. J, Burke E K and Abdullah S, (2009). An Extended Great Deluge Approach to the Examination Timetabling Problem, In proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications, Dublin, August 2009, pages 424-434

123. McCollum B, McMullan P, Burke E K, Parkes A J and Qu R, (2008). The second international timetabling competition: Examination timetabling track. Technical Report QUB/IEEE/Tech/ITC2007/Exam/v1.0/1. Queen's Belfast University, N. Ireland.

124. McCollum B, Schaerf A, Paechter B, McMullan P, Lewis R, Parkes A J, Di Gaspero L, Qu R and Burke E K, (2010). Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. INFORMS Journal on Computing 22(1), pp. 120-130.

125. McMullan P, (2007). An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, Lecture Notes in Computer Science, Springer, Vol 4487, pages 538-545.

126. Mehta N K, (1981). The application of a graph colouring method to an examination scheduling problem. Interfaces 11(5), pages 57-64.

127. Merlot L T G, Boland N, Hughes B D and Stuckey P J (2003). A hybrid algorithm for the examination timetabling problem. In: Burke, E.K. and De Causmaecker, P. (eds) (2003). Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference. Lecture Notes in Computer Science vol. 2740, Springer-Verlag. pp 207-231.

128. Meyers C and Orlin J B, (2006). Very large-scale neighborhood search techniques in timetabling problems, Lecture Notes in Computer Science (including subseries Lecture

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3867 LNCS, pp. 24-39.

129. MirHassani S A, (2006). Improving paper spread in examination timetables using integer programming, Applied Mathematics and Computation. Volume 179, Issue 2, , 15 August 2006, Pages 702-706.

130. Muller T, (2008). ITC 2007 solver description: A hybrid approach. In Practice and Theory of Automated Timetabling (PATAT 2008), pages 19–22.

131. Mushi A R, (2006). Tabu search heuristic for university course timetabling problem", African Journal of Science and Technology (AJST) Science and Engineering Series, Vol. 7, No. 1, pp. 34-40.

132. Nguyen Q V H, Ta Q B and Duong T A, (2005). A Memetic Algorithm for Timetabling" Proceedings of 3nd Int,Conf. RIVF'05 Research Informatics Vietnam-Francophony , 2005 , pages. 289 – 294.

133. Ong M L, Liew L H and Sim J (2009). Examination invigilation scheduling system in optimising lecturers' preference. In Proceedings of Conference on Scientific & Social Research (CSSR 08'09), Research Management Institute (RMI). 14-15 March 2009.

134. Petrovic S and Burke E K, (2004). University timetabling. Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapter 45, (Editor: J. Leung), CRC Press.

135. Pham D T and Karaboga D, (2000). Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks. 1st Ed., Springer, Berlin, Heidelberg, New York, London, ISBN: 1852330287, page: 302.

136. Pillay N and Banzhaf W, (2009). A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem, European Journal of Operational Research, Volume 197, Issue 2, 1 September 2009, Pages 482-491.

137. Pillay N, Banzhaf W, (2009). A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem, European Journal of Operational Research, Volume 197, Issue 2, 1 September 2009, Pages 482-491.

138. Pillay N, Banzhaf W, (2010). An informed genetic algorithm for the examination timetabling problem, Applied Soft Computing, Volume 10, Issue 2, March 2010, Pages 457-467.

139. Qu R and Burke E K, (2009). Hybridisations within a Graph Based Hyper-heuristic Framework for University Timetabling Problems" Journal of Operational Research Society, 60: Pages 1273-1285.

140. Qu R, Burke E K, McCollum B, (2009). Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems, European Journal of Operational Research, Volume 198, Issue 2, 16 October 2009, Pages 392-404.

141. Qu R, Burke E K, McCollum B, Merlot L T G and Lee S Y (2009). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. Journal of Scheduling, vol. 12, No 1, pp 55-89, 2009

142. Raghavjee R and Pillay N, (2008). An application of Genetic Algorithms to the School Timetabling Problem, Proccedings of SAICSIT '08, ACM Press, October 2008, pp. 193-199.

143. Rahman S A, Bargiela A, Burke E K, McCollum B and Özcan E, (2009). Construction of Examination Timetables Based on Ordering Heuristics, Accepted to ISCIS2009: 24th International Symposium on Computer and Information Sciences, Cyprus, 14-16 September 2009. pp. 680-685.

144. Reis L P and Oliveira E, (1999). A Constraint Logic Programming Approach to Examination Scheduling, Proc. Xth Irish Conference on Artificial Intelligence & Cognitive Science (AICS'99), Pages. 8-14.

145. Ross P and Corne D (1995). Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetabling Problems, Selected Papers from AISB Workshop on Evolutionary Computing, April 03-04, pages .94-102.

146. Ross P, Corne D and Fang H, (1994). Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation, Parallel Problem Solving from Nature III, Springer-Verlag, Heidelberg.

147. Rudová H and Murray K, (2003). University course timetabling with soft constraints. The Practice and Theory of Automated Timetabling IV: Selected Papers from 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT IV), Gent, Belgium, Lecture Notes in Computer Science 2740, Springer-Verlag. (Editors: E.K. Burke and P. De Causmaecker), pp 310-328.

148. Schaerf A, (1999). A survey of automated timetabling. Artificial Intelligence Review, 13(2), pp 87-127.

149. Silva D L and Obit J H, (2008). Great Deluge with Nonlinear Decay Rate for Solving Course Timetabling Problems. Proceedings of the 2008 IEEE Conference on Intelligent Systems (IS 2008), IEEE Press, pages 8.11-8.18,

150. Tassopoulos I X, Beligiannis G N, (2012). Solving effectively the school timetabling problem using particle swarm optimization, Expert Systems with Applications, Volume 39, Issue 5, Pages 6029-6040

151. Terashima M H, (1998). Combining of GAs and CSP strategies for solving the examination timetabling problem. PhD Thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, 1998.

152. Thompson J M and Dowsland K A (1996). General Cooling Schedules for a Simulated Annealing Based Timetabling System", In: Practice and Theory of Automated Timetabling, First International Conference: Selected Papers, (E. Burke & P. Ross , Eds.), Edinburgh, U.K., August/September 1995. Lecture Notes in Computer Science 1153, Springer-Verlag, 345-363.

153. Thompson J M and Dowsland K A, (1998). A robust simulated annealing based examination timetabling system, Computers & Operations Research, Volume 25, Issues 7-8, July 1998, pp 637-648.

154. Thompson J M and Dowsland K A, (2005). Ant Colony Optimisation for the examination scheduling problem, Journal of the Operational Research Society, 2005, 426-439, 56 (4).

155. Turabieh, H and Abdullah S, (2011). An integrated hybrid approach to the examination timetabling problem, Omega, Volume 39, Issue 6, December 2011, pages 598-607, ISSN 0305-0483, 10.1016/j.omega.2010.12.005.

156. Van den Broek J, Hurkens C and Woeginger G, (2009). Timetabling problems at the TU Eindhoven. European Journal of Operational Research 196 (3), pages 877–885.

157. White G M, Xie B S and Zonjic S. (2004). Using tabu search with longer-term memory and relaxation to create examination timetables, European Journal of Operational Research, Volume 153, Issue 1, Timetabling and Rostering, 16 February 2004, Pages 80-91.

158. White G.M and Xie B.S (2001). Examination timetables and tabu search with longer-term memory. The Practice and Theory of Automated Timetabling III: Selected Papers from 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT III), Konstanz, Germany, Lecture Notes in Computer Science 2079, Springer-Verlag. (Editors: E.K. Burke and W. Erben), pp 85-103.

159. Wilke P and Ostler J (2008). Solving the School Timetabling Problem Using Tabu Search, Simulated Annealing, Genetic and Branch & Bound Algorithms, Proceedings of PATAT 2008.

160. Wood D C, (1968). A system for computing University Examination Timetables, The Computer J. 11 (1968), pages 41-47.

161. Wren A, (1996). Scheduling, timetabling and rostering – A special relationship? The Practice and Theory of Automated Timetabling I: Selected Papers from 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT I), Edinburgh, UK, Lecture Notes in Computer Science 1153, Springer-Verlag. (Editors: E.K. Burke and P. Ross), Pages 46-75.

162. Wright M, (2001). Subcost-Guided Search - Experiments with Timetabling Problems, Journal of Heuristics, 7, pp 251-260, Kluwer Academic Publishers.

163. Zhang D F, Liu Y K, M'Hallah R and Leung S C H, (2010). A Simulated Annealing with a New Neighborhood Structure Based Algorithm for High School Timetabling Problems, European Journal of Operational Research, Vol. 203, No. 3, pages 550-558.

Appendix A: UMP examination data file format and specification

The Universiti Malaysia Pahang examination timetabling dataset contain 5 files:

1. Course file (e.g. sem10708-crs.txt)
2. Student file (e.g. sem10708-stu.txt)
3. Timeslot file (e.g. sem10708-tslot.txt)
4. Room file (e.g. sem10708-room.txt)
5. Room distance (e.g. sem10708-dist.txt)

The descriptions of the following files are as follows:

1. *Course file.* The course files contain information of the courses and total number of students (enrolments). The file is in the following format:

   <CourseCode> <Enrolment>
   BAA1312 148
   BAA2113 100
   BAA2513 128.

   The course data file is sorted in ascending order based on the <CourseCode>.

2. *Student files.* The student file listed the registered course of the particular students. This file is used to generate the conflict matrix. The file is in the following format:

   <StudentID> <Course>
   AA03002 BAA3223
   AA03002 BAA3412
   AA03002 BAA4513
   AA03003 BAA3223
   AA03003 BAA4223
   AA03003 BAA4513
   AA03030 BAA1312
   AA03030 BAA3032

   The student data file is sorted in ascending order based on the <studentID>.

3. *Timeslot file.* The timeslot files contain timeslot index, durations of timeslot (in minutes and penalty of a particular timeslot (if any). The file is in the following format:

   <TimeslotIndex> <Durations> <Penalty>
   1 180 0
   2 180 0
   3 180 0

4. *Room file.* The room files contain room code, room capacity and building code. The file is in the following format:

<RoomCode>	<RoomCapacity>	<BuildingCode>
DKU01	80	W
DKU02	80	W
WBK18	47	W

5. *Room Distance.* The files contain information of the room distance cost between the rooms. For example, the penalty cost between DKU01 and DKU02 is '001'.

<RoomCode1>	<RoomCode2>	<DistanceCost>
DKU01	DKU01	000
DKU02	DKU01	001
WBK18	DKU01	005

Appendix B: UMP Invigilation data file format and specification

The Universiti Malaysia Pahang examination timetabling dataset contain 5 files:

1. Staffs file (e.g. sem10708-staff.txt)
2. Invigilator-Room file (e.g. sem10708-invi.txt)
3. Lecturer own exam file (e.g. sem10708-lectEx.txt)

The descriptions of the following files are as follows:

1. *Staffs file.* The staff files contain information of the staff courses and the status (i.e. academic staffs or administration staffs). The staff ID with status = 0 is an administration staff while staff Id with status = 1 is an academic staff. The file is in the following format:

   <StaffID> <Status>
   0006 1
   0022 0
   0028 1

   The staff data file is sorted in ascending order based on the <StaffID>.

2. *Invigilator-Room files.* The invigilator-room file listed the required number of invigilators for a particular room. The room need to be assigned with the required number of invigilators. The file is in the following format:

   <RoomCode> <Invigilator required>
   DKU01 2
   DKU02 2
   WBK18 2

   The invigilator-room data file is sorted in ascending order based on the <RoomCode>.

3. *Lecturers own exam files.* The lecturers own exam files listed the course taught by the lecturers. The file is in the following format:

   <RoomCode> <LecturerID>
   BAA1312 0689
   BAA2113 0371
   BAA2713 0169

   The invigilator own exam data file is sorted in ascending order based on the <RoomCode>.

Appendix C: UMP semester1-200708 constructive result

No	Candidates list five (c = 5)						Candidates list one (c = 1)					
	LE	LD	LWD	SD (LE)	SD (LD)	SD (LWD)	LE	LD	LWD	SD (LE)	SD (LD)	SD (LWD)
1	4.74	8.16	5.81	7.54	7.24	8.33	10.98	19.34	13.27	14.81	16.88	15.05
2	6.67	8.79	6.39	6.82	7.23	8.12	15.75	14.54	15.19	13.96	16.04	14.91
3	7.05	8.32	5.64	6.18	7.21	6.22	11.75	13.10	14.06	14.95	16.25	18.77
4	5.45	8.30	6.17	5.28	5.99	6.63	18.02	14.61	14.52	18.07	16.67	15.92
5	6.13	6.58	5.67	5.89	6.99	6.00	13.99	14.88	15.12	16.11	18.72	16.18
6	6.61	6.94	5.51	6.57	5.76	6.39	17.29	15.31	14.56	16.06	13.78	16.96
7	4.99	7.65	8.29	6.32	7.51	5.97	16.50	15.40	15.81	12.86	16.72	17.53
8	5.79	8.23	6.17	6.72	8.26	7.29	14.92	16.20	14.88	13.79	14.37	14.84
9	5.78	7.17	5.76	7.38	6.16	6.74	15.92	17.07	20.66	15.87	17.61	17.79
10	5.69	7.12	6.16	7.32	7.29	7.10	11.55	15.60	16.94	15.32	17.89	16.00
11	5.36	5.99	5.59	7.98	7.97	6.12	15.05	13.71	15.39	17.27	16.11	16.19
12	5.55	9.55	7.67	6.52	7.82	7.10	16.17	14.14	15.92	16.01	16.22	16.29
13	6.66	6.30	5.67	7.50	6.65	6.16	15.84	14.67	15.46	15.77	16.27	15.44
14	5.33	6.75	6.24	7.41	5.96	7.67	15.89	15.11	13.58	17.62	15.20	18.63
15	5.29	6.78	6.36	7.10	8.18	9.35	18.43	17.96	19.44	19.37	15.97	16.75
16	6.71	7.33	5.56	8.13	7.18	5.84	13.31	15.21	14.63	15.38	13.21	14.33
17	5.77	7.34	5.84	6.44	6.42	6.19	15.93	17.30	12.69	16.07	16.72	15.01
18	7.15	7.52	5.96	6.80	7.22	8.31	14.15	16.51	18.89	13.98	17.46	14.41
19	6.57	9.04	5.42	6.71	6.61	8.99	13.65	13.72	13.89	16.94	15.60	18.22
20	5.05	8.49	5.83	7.24	8.26	7.29	18.41	15.84	17.08	16.56	18.54	18.71
21	7.63	7.91	5.80	7.37	8.61	6.47	16.35	15.45	14.47	15.51	15.46	15.70
22	5.57	7.40	5.91	6.32	6.23	5.84	15.85	17.36	14.72	18.45	15.83	15.62
23	5.91	7.26	5.30	8.49	6.45	5.81	13.58	17.11	16.22	19.03	19.39	16.19
24	4.88	8.71	7.18	5.97	6.38	6.11	14.80	15.77	17.99	20.74	17.47	17.23
25	6.42	8.48	5.90	6.55	6.03	6.21	13.92	17.35	20.70	17.86	17.94	14.50
26	6.12	7.41	5.63	6.56	7.56	6.89	12.37	17.07	14.23	17.85	15.66	14.63
27	5.48	6.30	7.03	7.77	6.44	7.97	17.07	15.71	17.27	15.88	16.35	16.73
28	5.90	8.12	6.02	7.27	8.72	7.64	20.03	13.04	15.08	16.60	14.85	15.04
29	5.98	9.28	5.92	7.84	8.51	6.75	16.20	17.07	16.36	17.92	15.51	17.63
30	4.97	8.04	7.43	7.51	8.64	7.57	14.52	17.95	14.95	18.02	17.27	15.52
31	6.63	8.34	5.36	7.34	5.76	9.55	19.57	17.59	12.91	19.35	17.83	14.57
32	7.98	9.64	5.81	6.58	7.11	7.08	14.07	14.52	17.14	15.25	15.67	16.87
33	5.53	11.12	6.30	6.50	6.65	7.10	16.16	14.67	19.02	17.20	15.60	15.93
34	6.11	7.73	5.30	6.45	6.93	6.12	16.53	16.82	14.22	13.26	15.16	18.79
35	6.59	8.74	6.23	6.34	6.77	6.78	15.84	17.03	16.09	14.68	13.87	15.68
36	7.66	7.96	6.83	6.29	6.92	6.88	13.81	15.88	15.86	14.05	16.57	14.00
37	7.34	7.60	6.73	6.43	6.77	9.78	16.79	15.41	16.88	14.17	17.72	17.90
38	6.28	8.31	5.36	7.85	6.51	6.28	15.57	15.90	16.09	17.77	14.07	17.57
39	6.14	7.38	7.08	6.92	7.31	5.68	15.32	14.93	17.76	12.66	14.24	16.53
40	6.40	7.15	5.05	7.14	8.16	5.49	15.30	18.53	16.09	14.59	17.50	15.93
41	5.44	7.81	6.76	6.63	7.96	7.02	19.17	18.28	14.94	13.76	14.14	19.20
42	5.67	8.71	6.81	7.52	8.11	6.75	13.44	16.78	17.44	16.38	18.59	14.77
43	6.06	7.14	6.35	6.27	7.49	8.49	14.57	17.20	15.72	16.96	16.79	17.14
44	5.49	8.85	5.87	7.79	7.66	6.58	19.28	14.78	18.07	16.71	13.11	20.41
45	6.16	7.78	6.04	6.48	8.51	7.31	18.50	16.00	17.92	14.68	16.90	13.97
46	6.65	7.69	5.92	7.54	7.89	6.01	17.13	16.35	11.43	16.96	17.00	16.31
47	5.50	8.25	5.30	7.60	6.19	7.26	12.46	15.11	16.21	13.93	15.68	15.02
48	6.31	6.82	5.41	6.48	7.13	7.32	13.70	14.86	15.53	15.91	17.78	15.09
49	6.69	6.48	6.24	6.95	7.47	6.89	14.95	15.67	14.89	15.96	13.12	14.34
50	5.07	7.05	5.92	7.50	8.11	6.52	15.15	17.23	13.00	15.76	15.10	17.87
Average	6.06	7.84	6.09	6.96	7.22	7.00	15.51	15.95	15.82	16.09	16.17	16.29
Var	0.57	0.96	0.44	0.43	0.70	1.05	4.40	2.02	3.90	3.26	2.35	2.37
Stdev	0.76	0.98	0.67	0.66	0.84	1.02	2.10	1.42	1.97	1.80	1.53	1.54
Min	4.74	5.99	5.05	5.28	5.76	5.49	10.98	13.04	11.43	12.66	13.11	13.97
Max	7.98	11.12	8.29	8.49	8.72	9.78	20.03	19.34	20.70	20.74	19.39	20.41

Appendix D: UMP semester1-200809 constructive result

No	Candidates list five ($c = 5$)						Candidates list one ($c = 1$)					
	LE	LD	LWD	SD (LE)	SD (LD)	SD (LWD)	LE	LD	LWD	SD (LE)	SD (LD)	SD (LWD)
1	7.72	9.56	7.22	7.51	9.29	8.79	16.50	16.27	19.73	20.47	18.02	16.12
2	8.72	10.07	6.71	10.44	10.96	9.05	17.60	18.26	15.58	14.71	20.54	18.79
3	7.82	10.45	7.76	10.54	7.79	8.48	15.69	17.11	16.79	17.87	18.11	16.18
4	7.60	9.33	7.79	8.18	10.05	8.63	15.66	18.43	14.65	17.57	19.51	18.49
5	8.44	9.51	8.13	8.03	9.86	9.67	17.09	17.71	15.96	20.45	16.71	16.91
6	9.20	9.79	7.61	10.67	9.04	9.07	20.66	17.37	17.77	18.34	20.11	16.82
7	7.21	9.18	6.98	8.86	10.51	10.39	15.87	17.34	16.50	19.44	18.39	18.68
8	8.51	10.15	8.15	10.56	9.83	9.13	17.39	17.53	18.43	18.10	19.30	16.76
9	7.73	9.17	7.09	10.12	9.26	9.43	18.70	18.94	19.03	16.87	17.26	19.94
10	7.04	8.45	9.45	8.27	10.37	7.58	19.63	17.72	17.89	18.59	17.02	18.33
11	9.25	9.65	8.07	8.68	10.29	7.51	14.74	15.16	17.56	19.66	17.05	17.29
12	9.14	10.48	6.75	9.59	10.37	8.65	15.20	20.41	15.34	20.13	18.18	20.33
13	7.86	9.65	9.39	9.71	9.43	10.14	18.25	17.42	16.83	15.09	17.26	15.79
14	9.23	8.58	7.93	9.58	8.70	8.06	17.82	16.75	16.76	17.09	20.58	20.74
15	8.25	10.82	6.76	10.34	9.27	9.11	16.83	19.16	14.95	16.11	15.79	16.65
16	8.87	9.30	8.09	9.76	8.52	9.49	18.18	16.74	18.92	16.89	19.52	19.10
17	9.37	9.19	7.61	8.88	9.79	8.78	18.81	18.73	16.82	15.00	17.61	18.41
18	8.06	9.29	7.52	9.76	9.58	10.22	19.65	17.97	17.56	17.23	17.40	17.35
19	7.65	9.36	8.86	11.77	9.56	8.13	18.50	20.85	20.08	15.00	18.12	16.71
20	7.06	11.23	7.59	9.56	9.68	9.73	16.93	18.65	16.64	18.30	18.86	17.87
21	7.96	9.75	7.56	8.01	7.29	8.14	16.85	17.13	18.13	17.93	16.93	16.08
22	9.40	10.24	8.75	9.58	8.89	9.27	15.25	16.30	16.01	20.75	21.10	18.83
23	8.78	9.50	8.30	7.87	11.15	8.07	16.39	17.29	18.34	16.51	20.67	15.37
24	7.98	10.95	6.97	10.22	9.33	9.08	15.73	18.95	21.66	18.32	15.11	18.71
25	7.29	8.61	7.49	9.25	9.71	9.49	17.25	21.09	19.26	19.57	15.87	18.84
26	7.34	12.03	8.48	11.42	9.65	8.09	16.92	18.29	17.14	17.65	18.63	16.54
27	8.12	11.05	8.15	9.44	8.28	9.35	17.87	20.11	17.17	18.97	19.12	17.60
28	7.95	11.39	8.14	9.56	12.69	8.41	17.51	16.94	18.09	18.35	19.74	17.74
29	8.08	8.92	9.69	9.15	9.38	8.67	19.97	17.53	16.93	20.44	19.90	19.08
30	7.88	8.55	8.26	8.78	10.81	8.48	17.43	14.48	16.36	19.28	16.21	19.48
31	9.16	10.87	7.30	7.69	10.97	8.65	18.38	17.14	19.23	15.50	19.19	17.29
32	7.24	9.72	7.41	10.25	9.64	10.56	15.82	22.69	17.03	15.67	16.70	18.31
33	7.82	8.47	8.28	10.39	8.74	8.54	16.93	16.81	18.97	17.02	17.04	21.09
34	8.64	8.55	8.90	9.20	10.07	9.27	18.41	16.52	19.31	19.72	16.05	16.37
35	7.10	9.84	8.28	9.01	9.13	8.03	15.61	16.81	16.27	17.32	18.27	18.23
36	8.68	10.10	8.23	9.22	9.96	9.33	17.14	17.21	16.20	19.82	16.14	18.21
37	7.46	8.79	9.07	8.59	9.56	8.81	15.85	15.60	15.73	19.37	16.27	17.79
38	8.62	9.40	7.47	9.07	9.01	9.72	18.44	18.04	17.94	18.24	19.63	17.61
39	8.09	11.14	7.64	8.31	9.81	8.31	19.53	18.51	16.23	20.41	16.99	18.03
40	8.05	8.93	6.61	9.57	9.65	8.43	20.11	17.60	14.38	20.10	17.69	15.74
41	7.17	9.49	7.77	9.24	9.57	8.75	23.11	18.52	17.19	17.69	18.46	19.61
42	8.47	9.07	7.64	9.51	9.36	8.84	16.92	19.33	15.68	19.26	17.62	17.09
43	8.48	9.17	7.68	8.98	8.79	9.05	15.24	19.19	17.66	14.64	18.57	15.41
44	7.75	11.99	6.93	10.18	9.73	9.44	17.77	20.74	17.17	16.37	16.14	16.36
45	7.97	11.03	7.65	8.32	10.17	8.45	16.76	19.18	15.45	18.61	19.74	16.67
46	7.28	9.02	8.33	9.33	10.12	7.92	17.01	17.31	13.89	19.20	16.36	19.11
47	8.50	9.65	7.84	10.96	9.43	8.36	18.28	19.02	19.45	18.96	17.73	16.97
48	7.49	10.26	7.79	9.59	9.06	8.37	17.88	16.73	16.06	17.65	19.21	19.07
49	8.32	8.32	8.10	10.43	9.29	8.74	16.57	17.26	21.45	18.40	19.96	17.09
50	7.46	7.47	7.94	10.14	11.00	7.00	17.37	16.52	18.29	18.91	15.27	15.80
Average	8.11	9.71	7.88	9.44	9.65	8.83	17.48	17.95	17.33	18.07	18.03	17.75
Var	0.46	0.98	0.50	0.91	0.79	0.55	2.60	2.44	2.86	2.83	2.40	1.97
Stdev	0.68	0.99	0.71	0.95	0.89	0.74	1.61	1.56	1.69	1.68	1.55	1.40
Min	7.04	7.47	6.61	7.51	7.29	7.00	14.74	14.48	13.89	14.64	15.11	15.37
Max	9.40	12.03	9.69	11.77	12.69	10.56	23.11	22.69	21.66	20.75	21.10	21.09

Appendix E: UMP semester1-200708 *modified*-GDA results

No.	Nh1- 1500 iterations				Nh1 – 3000 iterations				Nh2 – 1500 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
1	5.84	5.95	5.55	5.1	6.58	5.56	4.65	4.24	16.68	13.54	10.3	7.82
2	5.98	6.47	5.04	5.1	5.39	4.95	5.06	4.49	16.68	13.54	10.3	7.82
3	6.41	6.1	5.39	4.81	6.19	5.45	4.88	4.64	16.68	13.57	10.3	7.82
4	6.46	5.89	5.14	5.15	5.44	5.46	4.77	4.58	16.68	13.54	10.3	7.82
5	6.77	6.18	5.61	5.23	5.12	5.75	5.2	4.48	16.68	13.54	10.3	7.82
6	7.18	6.22	5.22	5.14	6.00	5.26	5.21	4.28	16.68	13.5	10.3	7.82
7	6.3	6.66	5.17	4.79	5.56	5.7	4.58	4.61	16.68	13.5	10.3	7.82
8	6.33	6.18	4.8	5.17	5.92	5.51	5.18	4.53	16.68	13.53	10.3	7.82
9	6.44	6.13	5.13	5.21	5.84	5.47	5.2	4.25	16.68	13.53	10.3	7.82
10	7.03	5.84	5.4	4.9	5.87	5.78	4.62	4.63	16.68	13.53	10.3	7.82
11	6.72	6.1	5.85	4.6	5.72	5.43	4.41	4.32	16.68	13.52	10.3	7.82
12	6.13	6.03	5.76	4.82	6.00	5.38	5.04	4.58	16.68	13.52	10.3	7.82
13	6.68	5.97	5.33	4.74	5.40	5.71	4.7	4.61	16.68	13.53	10.3	7.82
14	6.7	6.25	5.5	5.04	5.64	5.39	4.88	4.31	16.68	13.54	10.3	7.82
15	6.52	6.18	5.31	5.4	5.09	5.73	5.08	4.53	16.68	13.54	10.3	7.82
16	6.63	5.94	5.64	5.04	5.63	5.65	4.99	4.57	16.68	13.53	10.3	7.82
17	6.07	6.3	4.93	4.6	5.51	5.56	5.18	4.43	16.68	13.46	10.3	7.82
18	6.63	6.28	5.03	4.72	6.01	5.41	5.04	4.43	16.68	13.52	10.3	7.82
19	6.06	5.7	5.83	5.02	5.98	5.32	5.12	4.98	16.68	13.52	10.3	7.82
20	6.54	5.9	5.23	4.8	6.27	5.46	4.88	4.6	16.68	13.52	10.3	7.82
21	6.53	5.8	5.12	5.16	6.26	5.24	5.07	4.66	16.68	13.52	10.3	7.82
22	6.78	5.98	4.97	5.06	5.80	5.18	4.78	4.73	16.68	13.54	10.3	7.82
23	6.68	6.64	5.56	5.04	5.83	5.34	5.29	4.81	16.68	13.52	10.3	7.82
24	6.82	6.19	5.1	5.17	5.78	6.12	4.99	4.71	16.68	13.53	10.3	7.82
25	7.03	6.58	5.16	5.43	6.23	5.78	5.24	4.36	16.68	13.53	10.3	7.82
26	6.41	6.25	5.54	5.1	6.14	5.78	4.92	4.89	16.68	13.54	10.3	7.82
27	6.86	6.75	4.98	5.02	5.81	5.46	4.93	4.65	16.68	13.52	10.3	7.82
28	6.64	6.24	5.15	4.99	5.79	5.48	4.99	4.75	16.68	13.52	10.3	7.82
29	7.16	6.04	5.31	5.18	6.46	5.13	5.13	4.7	16.68	13.53	10.3	7.82
30	6.45	6.69	5.21	5.1	6.02	5.52	4.96	4.63	16.68	13.52	10.3	7.82
31	6.51	6.61	5.16	5.02	5.42	5.63	4.86	4.46	16.68	13.53	10.3	7.82
32	6.52	5.77	5.83	5.01	6.40	5.83	5.17	4.52	16.68	13.54	10.3	7.82
33	6.98	5.68	5.52	5.09	5.82	5.34	5.03	4.88	16.68	13.53	10.3	7.82
34	7.02	6.2	5.06	5.22	5.52	5.2	5.35	4.7	16.68	13.52	10.3	7.82
35	6.33	6.38	5.69	5.39	5.87	5.97	4.76	4.89	16.68	13.52	10.3	7.82
36	6.94	6.18	5.87	5.3	6.03	5.98	4.49	4.68	16.68	13.54	10.3	7.82
37	6.7	6.77	5.59	4.79	5.93	5.87	5.17	4.55	16.68	13.51	10.3	7.82
38	7.1	6.35	5.64	4.89	6.60	5.78	4.77	4.83	16.68	13.54	10.3	7.82
39	7.24	6.85	5.34	4.77	5.94	5.3	4.79	4.7	16.68	13.54	10.3	7.82
40	6.7	6.11	5.83	5.24	5.63	5.7	4.89	4.96	16.68	13.48	10.3	7.82
41	6.6	5.88	5.38	5.19	6.27	5.24	4.76	4.78	16.68	13.52	10.3	7.82
42	6.87	6.28	6.13	4.97	6.21	5.75	4.73	4.77	16.68	13.53	10.3	7.82
43	7.03	6.79	5.69	5.07	5.37	5.88	4.82	4.98	16.68	13.53	10.3	7.82
44	6.4	5.99	6.08	5.37	6.15	5.3	5.62	4.46	16.68	13.52	10.3	7.82
45	6.33	6.33	5.89	4.99	5.85	5.87	5.2	4.76	16.68	13.54	10.3	7.82
46	6.83	6.56	5.5	4.82	6.23	5.14	4.76	4.87	16.68	13.54	10.3	7.82
47	6.99	6.44	5.74	5.37	5.82	5.25	4.82	4.74	16.68	13.52	10.3	7.82
48	5.71	6.53	5.71	5.63	6.14	5.37	5.33	4.34	16.68	13.51	10.3	7.82
49	6.85	6.61	5.77	5.12	5.52	5.36	5.27	4.73	16.68	13.52	10.3	7.82
50	6.85	6.84	5.68	5.29	5.67	5.57	5.22	4.36	16.68	13.54	10.3	7.82
Min	5.71	5.68	4.80	4.60	5.09	4.95	4.41	4.24	16.68	13.46	10.30	7.82
Max	7.24	6.85	6.13	5.63	6.60	6.12	5.62	4.98	16.68	13.57	10.30	7.82
Ave	6.63	6.25	5.44	5.06	5.87	5.53	4.98	4.62	16.68	13.53	10.30	7.82
Stdev	0.35	0.32	0.32	0.22	0.35	0.26	0.24	0.20	0.00	0.02	0.00	0.00

Appendix E: UMP semester1-200708 *modified*-GDA results (*cont...*)

No.	Nh2 – 3000 iterations				Nh3- 1500 iterations				Nh3 – 3000 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
1	16.68	13.52	10.3	7.82	7.69	7.14	5.68	6.02	7.43	6.88	5.67	6.09
2	16.68	13.53	10.3	7.82	7.24	7.3	5.09	5.84	6.78	6.63	5.26	5.93
3	16.68	13.52	10.3	7.82	8.74	7.16	8.27	5.93	8.18	6.90	8.04	5.96
4	16.68	13.54	10.3	7.82	8.71	7.1	5.22	6.06	6.73	6.49	7.35	5.71
5	16.68	13.54	10.3	7.82	6.91	6.84	9.13	6.26	6.36	6.82	6.85	6.08
6	16.68	13.51	10.3	7.82	7.13	6.95	6.97	5.85	8.16	7.00	5.67	5.56
7	16.68	13.53	10.3	7.82	8.03	6.98	5.84	5.94	6.94	7.17	6.28	5.51
8	16.68	13.53	10.3	7.82	8.46	7.33	5.65	5.9	6.88	6.50	6.2	5.97
9	16.68	13.53	10.3	7.82	8.35	6.41	6.62	5.11	6.66	6.62	7.85	5.45
10	16.61	13.53	10.3	7.82	7.51	6.99	6.17	5.57	7.02	7.38	6.08	6.09
11	16.68	13.52	10.3	7.82	10.61	6.66	6.79	6.02	7.54	6.84	5.99	5.78
12	16.68	13.53	10.3	7.82	7.1	7.07	5.9	6.1	8.23	7.06	5.41	5.31
13	16.68	13.53	10.3	7.82	9.36	7.17	7.94	5.31	8.81	6.45	6.21	5.67
14	16.68	13.55	10.3	7.82	8.04	6.95	5.39	6.13	6.88	6.61	5.76	5.69
15	16.68	13.52	10.3	7.82	7.39	7.02	5.82	5.57	8.73	7.03	7.99	5.83
16	16.68	13.53	10.3	7.82	8.2	7.31	5.84	5.92	9.37	7.05	5.37	5.75
17	16.68	13.55	10.3	7.82	9.18	7.37	8.53	5.96	6.83	7.14	6.65	5.84
18	16.68	13.54	10.3	7.82	7.38	7.17	5.46	6.07	6.85	7.09	5.63	5.47
19	16.68	13.53	10.3	7.82	10.3	7.23	7.76	6.08	7.17	7.20	5.66	6.22
20	16.68	13.52	10.3	7.82	7.7	7.12	6.57	6.62	8.65	6.78	7.86	5.07
21	16.68	13.52	10.3	7.82	8.05	7.31	5.98	5.38	6.61	6.71	6.26	5.47
22	16.68	13.52	10.3	7.82	9.16	6.9	6.82	6.33	7.04	6.94	5.66	5.89
23	16.68	13.53	10.3	7.82	7.96	6.57	5.27	6.62	8.49	6.22	5.57	5.85
24	16.68	13.53	10.3	7.82	7.36	7.11	6.23	5.83	7.26	6.70	5.22	5.34
25	16.61	13.52	10.3	7.82	9.93	6.88	7.14	5.76	7.49	6.88	5.64	5.83
26	16.68	13.52	10.3	7.82	7.44	6.45	6.21	6.12	7.06	6.62	6.9	5.75
27	16.68	13.53	10.3	7.82	6.9	6.62	6.08	5.89	7.22	7.02	6.21	5.75
28	16.68	13.53	10.3	7.82	8.28	7.22	5.64	6.12	9.58	6.91	5.33	6.02
29	16.68	13.54	10.3	7.82	9.35	7.45	6.19	5.94	6.76	6.95	7.4	5.78
30	16.68	13.53	10.3	7.82	9.39	6.94	5.88	6.37	6.33	6.53	5.69	5.74
31	16.68	13.53	10.3	7.82	8.19	8.01	5.81	6.03	7.4	6.49	7.79	5.94
32	16.68	13.53	10.3	7.82	6.58	6.78	5.7	6.04	6.66	6.86	5.64	5.98
33	16.68	13.53	10.3	7.82	10.39	7.03	6.32	5.93	7.1	7.32	5.55	5.78
34	16.68	13.52	10.3	7.82	9.09	7.25	5.57	6.07	6.87	6.18	7.75	5.72
35	16.68	13.53	10.3	7.82	7.43	7.1	6.18	6.01	7.75	6.69	6.85	5.94
36	16.68	13.53	10.3	7.82	7.73	7.44	5.64	5.65	6.98	6.39	5.95	5.62
37	16.68	13.52	10.3	7.82	8.13	7.23	6.18	6.58	6.96	7.05	5.75	6.31
38	16.68	13.53	10.3	7.82	7.27	7.21	7.98	6.3	7.12	6.87	6.27	5.49
39	16.68	13.52	10.3	7.82	9.49	6.83	6.16	6.31	6.7	7.05	5.35	5.56
40	16.68	13.52	10.3	7.82	7.71	7.08	7.98	5.62	7.17	6.73	5.53	5.94
41	16.68	13.52	10.3	7.82	7.07	6.86	6.49	5.58	7.56	6.70	5.66	5.86
42	16.68	13.53	10.3	7.82	7.8	6.94	5.84	5.91	7.27	6.85	5.71	5.92
43	16.68	13.51	10.3	7.82	7.19	7.03	6.01	5.81	7.37	6.96	5.7	5.54
44	16.68	13.52	10.3	7.82	7.42	6.6	7.16	5.93	7	6.93	6.4	5.88
45	16.68	13.52	10.3	7.82	8.97	6.89	6.31	5.94	8.18	6.74	5.19	5.95
46	16.68	13.51	10.3	7.82	6.92	6.85	7.42	6.02	6.62	7.32	5.13	5.56
47	16.68	13.53	10.3	7.82	8.38	7.21	6.06	5.94	6.54	6.86	5.96	5.49
48	16.68	13.53	10.3	7.82	8.39	7.71	5.73	5.92	6.26	7.11	6.34	6.22
49	16.68	13.53	10.3	7.82	9.28	6.44	5.22	5.9	8.51	6.33	5.11	6.05
50	16.68	13.53	10.3	7.82	7.72	6.96	5.88	5.77	7.36	6.85	5.13	5.02
Min	16.61	13.51	10.30	7.82	6.58	6.41	5.09	5.11	6.26	6.18	5.11	5.02
Max	16.68	13.55	10.30	7.82	10.61	8.01	9.13	6.62	9.58	7.38	8.04	6.31
Ave	16.68	13.53	10.30	7.82	8.18	7.04	6.35	5.96	7.35	6.83	6.13	5.76
Stdev	0.01	0.01	0.00	0.00	1.00	0.31	0.93	0.30	0.78	0.28	0.85	0.27

Appendix E: UMP semester1-200708 *modified*-GDA results (*cont...*)

No.	Nh4 – 1500 iterations				Nh4 – 3000 iterations				Nh6-1500 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
1	9.93	10.46	7.26	6.18	9.69	8.79	6.96	6.25	11.97	12.02	9.33	6.57
2	9.59	10.68	7.11	6.08	9.38	10.23	6.99	6.37	11.53	11.71	9.47	7.02
3	10.06	10.83	7.02	6.51	10.64	9.42	6.8	6.06	13.4	11.33	9.47	7.00
4	10.52	10.68	6.92	6.3	10.07	9.57	7.44	6.25	13.02	11.59	9.47	7.00
5	10.4	11.18	7.1	6.41	9.85	9.72	6.98	6.08	13.22	11.53	9.33	7.02
6	10.02	10.72	8.28	6.59	10.69	10.05	7.1	5.9	13.23	11.77	9.47	7.00
7	10.16	11.02	6.96	6.48	9.45	9.99	7.03	6.14	11.64	11.55	9.33	7.00
8	10.02	11.27	7.91	6.53	10.35	9.55	6.89	6.16	11.31	11.54	9.33	7.00
9	10.38	11.41	8.87	6.22	10.13	9.86	6.76	6.12	11.5	11.51	9.33	7.02
10	11.22	11.11	7.45	6.14	9.94	9.7	7.62	6.45	11.66	11.32	9.33	7.00
11	9.77	11.32	6.96	6.21	10.25	10.01	6.96	6.12	11.56	11.48	9.33	7.00
12	9.4	10.61	7.54	6.33	10.44	10.36	6.73	6.16	11.52	11.95	9.47	7.02
13	10.04	11.06	6.93	6.41	10	10.07	7.29	6.26	11.66	11.46	9.47	7.00
14	10.03	11.45	6.72	6.39	9.58	9.72	7.2	6.45	11.41	12	9.33	7.02
15	10.18	10.68	7.14	6.28	10.01	9.76	7.21	6.39	11.83	11.35	9.47	7.02
16	10.44	11.04	7.7	6.3	10.27	9.65	7.12	6.3	11.32	11.3	9.33	7.00
17	10.15	10.93	7.53	6.38	10.5	9.89	6.75	6.32	11.69	11.83	9.33	7.02
18	10.46	10.9	7.78	6.61	10.34	9.65	7.63	6.33	11.19	11.79	9.33	7.00
19	11.34	10.98	7.54	6.67	11.92	9.45	7.37	6.2	12.99	11.53	9.47	6.96
20	10.16	11.43	7.35	6.63	10.51	10.87	7.78	6.29	11.99	11.52	9.33	7.00
21	10.08	10.94	7.19	6.47	10.22	9.07	7.06	6.2	13.17	11.56	9.33	7.08
22	10.36	10.93	6.52	6.71	9.9	9.84	7.46	6.21	11.97	11.91	9.47	6.96
23	9.81	10.79	6.77	6.94	10.85	10.15	7.11	6.91	11.24	11.8	9.33	6.76
24	10.06	11.19	7.7	6.31	10.28	10.03	6.59	5.96	11.64	11.59	9.33	7.00
25	11.18	11.97	7.98	6.1	9.96	9.68	7.06	6.48	13.02	11.25	9.33	7.00
26	9.82	11.41	7.56	6.47	8.93	10.37	6.93	6.39	12.99	11.47	9.33	7.00
27	10.05	11.19	7.55	6.75	8.87	9.73	7.04	6.15	13.4	12.01	9.33	7.00
28	11.59	10.68	6.98	6.54	9.71	9.87	6.57	6.09	11.41	11.54	9.33	7.00
29	11.3	10.38	6.87	7.07	9.32	10.04	7.69	6.07	11.62	11.9	9.47	7.00
30	10.07	11.24	7.12	6.38	10.19	10.13	6.69	6.19	11.83	12.07	9.47	7.00
31	10.63	11.16	7.03	6.23	9.68	9.51	6.47	6.44	11.27	11.63	9.33	7.02
32	10.51	10.66	7.46	6.69	10.23	9.96	7.3	6.51	11.77	11.53	9.33	7.00
33	9.85	10.97	7.93	6.55	9.69	10.89	6.84	6.28	12.11	11.48	9.33	7.00
34	11.32	10.7	7.52	6.39	9.11	10.15	7.2	6.36	12.99	11.39	9.33	7.00
35	11.39	10.64	7.28	6.47	10.17	10.6	6.83	6.39	11.34	11.78	9.33	7.02
36	10.13	10.84	7.61	6.39	9.75	10.44	7.17	6.09	13.02	11.79	9.33	7.02
37	9.86	11.13	7.92	6.47	10.01	10.3	7.32	6.35	13.17	11.64	9.47	7.00
38	10.61	11.01	7.58	6.31	10.86	10.34	6.79	6.36	11.25	11.46	9.33	7.00
39	10.49	11.29	7.18	6.85	10.35	10.17	7.1	6.65	11.57	11.74	9.33	7.02
40	9.9	10.88	6.61	6.5	9.95	10.78	7.06	6.29	13.43	11.36	9.33	7.02
41	10.26	11.13	7.08	6.68	9.62	10.3	7.25	6.15	13.4	11.55	9.33	7.00
42	10.49	10.61	7.38	6.57	9.36	10.21	6.85	6.31	11.57	11.97	9.33	7.02
43	10	11.22	7.28	6.47	10.32	10.16	6.95	6.36	13.23	11.6	9.33	7.00
44	10.37	11.83	7.33	6.38	10.82	10.24	7.61	6.31	11.34	11.65	9.33	7.02
45	10.32	10.63	6.95	6.35	10.3	10.05	7.38	6.27	11.41	11.32	9.47	7.02
46	9.91	10.88	6.91	6.53	9.93	9.98	7.12	6.05	11.59	11.7	9.47	7.00
47	9.64	11.13	7.43	6.66	10.17	10.69	6.89	6.62	12.99	11.68	9.33	7.08
48	11.22	10.71	6.9	6.4	10.5	10.25	7.24	6.27	12.99	11.63	9.33	7.02
49	11.96	11.16	7.03	6.42	9.78	10.62	6.99	6.54	13.4	11.6	9.33	6.96
50	9.7	10.77	7.31	6.44	10.48	10.79	7.36	6.08	11.57	11.76	9.33	7.00
Min	9.40	10.38	6.52	6.08	8.87	8.79	6.47	5.90	11.19	11.25	9.33	6.57
Max	11.96	11.97	8.87	7.07	11.92	10.89	7.78	6.91	13.43	12.07	9.47	7.08
Ave	10.34	11.00	7.32	6.46	10.07	10.03	7.09	6.28	12.15	11.63	9.37	6.99
Stdev	0.57	0.32	0.44	0.20	0.54	0.44	0.30	0.18	0.80	0.21	0.06	0.07

Appendix E: UMP semester1-200708 *modified*-GDA results (*cont...*)

No.	Nh6 – 3000 iterations				Nh7 – 1500 iterations				Nh7-3000 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
1	13.17	11.69	9.33	7.02	7.49	10.69	6.31	5.2	7.06	5.9	5.63	4.14
2	11.74	11.69	9.47	7.02	9.37	11.47	5.91	4.97	8.07	7.79	4.89	4.99
3	13.4	11.79	9.47	7.00	8.84	12.21	6.17	5.19	7.52	7.27	4.94	4.52
4	11.78	11.66	9.47	7.00	9.08	11.86	5.89	5.25	8.42	7.84	5.09	4.21
5	11.68	11.56	9.47	6.96	7.79	11.28	5.2	5.51	8.49	6.41	4.9	4.33
6	11.41	10.86	9.47	7.00	9.38	10.45	5.35	5.02	8.97	7.97	5.32	4.29
7	11.8	11.5	9.33	7.00	9.97	9.44	5.59	4.95	7.73	8.26	5.86	4.82
8	13.43	11.64	9.33	7.00	10.34	10.31	6.05	5.09	8.8	8.08	6	4.84
9	13.02	11.85	9.33	7.08	11.25	11.29	5.58	5.38	9.77	9.25	4.54	4.68
10	11.73	12	9.47	7.00	12.97	10.57	5.67	5.83	10.64	9.38	5.63	4.78
11	13.43	11.58	9.33	6.96	12.14	10.46	5.82	6.26	9.61	9.6	5.62	4.81
12	11.3	11.7	9.47	6.41	11.62	10.5	6	5.66	10.33	9.91	6.34	5.13
13	13.23	11.8	9.33	7.02	12.56	11.25	6.1	5.66	9.51	9.35	6.11	4.64
14	11.57	11.58	9.47	7.00	12.15	11.7	6.07	5.5	10.57	8.54	5.31	4.53
15	11.24	11.5	9.33	7.02	11.66	11.3	6.55	5.39	10.58	8.86	5.11	5.13
16	13.4	11.58	9.47	7.00	11.54	10.78	6.29	5.22	10.47	9.04	5.88	5.02
17	11.66	11.5	9.47	7.00	11.42	10.38	6.53	5.96	10.84	9.65	5.11	4.82
18	11.16	11.4	9.47	7.00	12.63	12.22	6.91	5.68	12.41	9.38	5.53	4.78
19	13.17	11.7	9.47	6.96	9.83	11.6	6.22	5.93	11.85	11.07	5.32	5.29
20	11.68	11.75	9.33	7.00	11.55	10.01	5.81	5.68	13.57	10.51	6.03	5.5
21	11.63	11.5	9.33	7.00	13.09	11.59	6.83	5.29	10.43	9.59	5.31	5.2
22	11.64	11.76	9.33	7.02	11.47	11.16	6.78	4.95	12.66	9.54	5.4	4.79
23	13.4	11.53	9.47	7.00	12.29	11.1	6	5.53	11.79	10.29	6.29	4.42
24	11.7	11.5	9.33	7.02	13.42	10.55	5.87	5.51	12.43	9.94	5.8	4.5
25	11.2	11.56	9.47	7.02	12.48	10.84	5.71	5.48	12.25	7.44	6.13	4.37
26	13.23	11.51	9.33	6.96	12.69	11.46	7.06	5.26	7.45	10.56	5.46	5.29
27	13.23	11.43	9.47	7.00	12.21	11.4	6.35	4.93	7.83	10.65	5.52	4.84
28	11.56	11.65	9.33	7.00	13.54	10.99	6.63	5.27	8.15	9.93	6.42	5.43
29	11.64	11.32	9.47	7.00	12.93	10.38	5.73	5.48	7.97	10.53	5.34	5.07
30	13.43	11.39	9.33	7.00	11.26	10.71	6.41	5.49	7.52	10.25	5.72	5.41
31	13.4	11.09	9.33	6.96	12.66	11.15	7.28	5.68	11.01	8.12	5.11	4.89
32	13.4	11.54	9.33	7.00	14.05	10.83	6.67	5.67	8.78	7.68	5.53	4.77
33	13.4	11.65	9.47	7.00	10.75	10.81	5.69	5.06	8.71	7.65	5.59	4.99
34	13.17	11.85	9.33	7.00	14.12	11.46	6.63	4.61	9.87	7.78	5.36	5
35	13.4	11.58	9.33	7.00	12.15	9.87	6.73	5.67	9.58	8.6	5.36	5.02
36	13.22	11.97	9.47	6.21	11.25	10.44	6.9	5.23	9.09	7.79	5.71	4.88
37	12.06	11.61	9.47	7.00	15.26	10.31	6.93	4.7	11.1	9.42	5.67	4.27
38	11.37	11.74	9.33	7.00	13.95	10.97	6.07	6.09	10.3	9.42	5.85	5.15
39	11.85	11.56	9.47	7.02	11.34	11.73	6.51	5.56	13.17	9.27	6.46	5.16
40	12.99	11.91	9.33	7.12	11.75	11.97	6.09	5.32	12.21	8.38	5.93	5.44
41	11.55	11.84	9.33	7.00	11.7	10.94	5.91	5.46	11.12	8.97	5.08	5.25
42	11.67	11.44	9.33	7.00	11.59	10.49	6.25	5.08	10.01	8.51	5.59	4.85
43	12.99	12.03	9.47	7.02	14.39	11.57	6.16	5.34	11.65	8.09	6.32	5.6
44	11.66	12.37	9.33	7.00	14.16	10.59	6.84	4.98	13	8.3	5.77	5.6
45	11.7	11.24	9.47	7.00	14.74	10.72	6.48	5.68	10.92	9.52	5.15	4.9
46	11.68	11.46	9.33	7.02	12.94	11.72	6.4	5.48	13.4	9.6	5.74	4.73
47	11.41	11.39	9.47	6.96	12.27	12.29	5.8	5.13	13.93	10.49	5.4	4.82
48	13.22	11.58	9.33	7.00	12.56	11.67	5.7	5.45	13.19	9.88	4.81	4.78
49	13.4	11.46	9.47	7.00	13.22	11.96	6.6	5.45	12.37	10.33	5.96	4.88
50	11.72	11.72	9.47	7.00	11.56	11.3	6.65	5.09	12.62	9.88	5.43	4.52
Min	11.16	10.86	9.33	6.21	7.49	9.44	5.20	4.61	7.06	5.90	4.54	4.14
Max	13.43	12.37	9.47	7.12	15.26	12.29	7.28	6.26	13.93	11.07	6.46	5.60
Ave	12.34	11.61	9.40	6.98	11.91	11.05	6.23	5.39	10.39	9.01	5.57	4.88
Stdev	0.86	0.24	0.07	0.14	1.69	0.64	0.47	0.34	1.91	1.15	0.44	0.36

Appendix E: UMP semester1-200708 *modified*-GDA results (*cont...*)

No.	Nh8 – 1500 iterations				Nh8 – 3000 iterations				Nh9-1500 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
1	8.21	7.91	7.04	6.18	7.9	8.15	6.94	6.28	9.41	9.15	6.17	5.79
2	8.55	8.56	7.06	6.37	8.04	8.39	6.66	6.08	7.94	7.57	6.26	5.92
3	7.86	8.3	7.3	6.37	8.26	8.58	7.27	5.77	8.72	8.38	6.03	6.37
4	8.2	8.49	7.24	6.44	9.59	9.21	7.06	6.06	7.81	8.74	7.3	6.32
5	8.85	8.67	7.34	6.75	8.39	7.92	7.06	7	8.46	7.41	6.36	6.05
6	8.74	7.77	6.88	6.37	8.98	8.06	7.03	6.25	7.65	8.52	6.42	6.17
7	8.76	9.01	6.7	6.37	8.21	8.35	6.81	6.37	7.86	8.12	7.04	6.19
8	8.05	8.01	6.89	6.22	8.45	8.34	6.67	6.39	7.7	7.79	6.12	6.31
9	8.12	8.73	6.71	6.3	8.42	8.85	7.1	6.59	8.6	8.08	6.09	6.37
10	8.51	8.72	7.75	6.37	8.79	8.77	7.25	6.42	8.27	8.42	6.51	6.06
11	9.08	8.37	7.1	6.75	8.27	7.73	7.22	6.75	8.42	8.15	6.04	5.97
12	7.97	8.81	7.1	6.43	7.73	7.99	6.97	6.29	7.75	8.31	5.73	5.78
13	8.89	8.91	7.13	6.44	8.57	8.91	7.16	6.44	7.95	7.5	6.3	5.45
14	8.13	8.2	7.82	7.13	7.4	7.77	7.44	6.66	7.53	7.27	5.97	5.98
15	8.75	8.53	7.44	6.5	7.92	8.31	7.19	6.75	7.89	7.86	6.18	5.81
16	8.37	8.24	6.65	6.56	8.82	8.62	7.83	6.69	7.87	7	5.98	5.8
17	7.68	8.81	7.17	6.66	7.8	8.43	6.7	6.37	7.32	7.69	6.14	5.83
18	8.29	8.55	7.02	6.75	8.32	8.78	7.15	5.84	8.1	9.15	6.76	5.86
19	8.54	8.26	6.71	6.37	8.03	9.12	7.35	6.75	7.91	6.91	6.06	6.59
20	8.66	8.22	7.1	6.75	8.16	7.76	6.92	6.28	8.92	8.84	6.33	6.04
21	8.59	8.74	7.27	6.75	7.73	8.29	7.23	6.06	8.08	8.09	5.73	6.02
22	8.14	8.02	7.21	6.88	8.16	8.74	7.18	6.56	8.22	8.32	5.43	5.88
23	8.52	8.56	7.55	6.22	9.11	9.17	6.98	6.88	8.45	8.84	5.81	5.89
24	9.48	9.18	7.14	6.37	8.81	8.09	7.59	6.37	7.9	7.32	6.91	5.88
25	9.11	7.76	7.28	6.75	8.96	7.98	7.05	6.66	7.58	7.87	6.61	5.73
26	8.54	8.31	7.05	6.48	8.79	8.09	7.42	6.66	7.75	8.26	6.08	6.72
27	7.86	8.42	7.02	6.28	8.48	8.34	7.65	6.44	7.53	7.64	6.5	5.72
28	8.04	9.37	6.64	6.75	8.1	8.7	7.62	6.75	8.29	8.32	5.5	5.97
29	8.53	8.21	7.81	6.75	8.55	8.34	7.02	6.37	7.68	9.3	6.15	6.49
30	8.47	7.72	7.39	6.75	8.45	8.17	6.81	6.75	8.93	8.11	6.54	6.4
31	8.51	9.27	6.96	6.75	8.28	8.37	6.71	6.75	8.54	7.18	6.41	6.61
32	9.06	8.35	7.7	6.44	8.26	8.2	6.96	6.28	7.75	8.33	6.54	6.17
33	8.69	8.38	7.45	6.75	8.49	8.26	6.8	6.66	8.27	7.51	6.86	6.17
34	8.51	7.87	6.84	6.37	7.87	7.92	7.46	6.28	8.47	7.16	6.18	6.15
35	8.88	8.71	7.1	6.39	8.14	8.91	6.97	6.75	8.23	8.66	7.02	5.71
36	8.49	8.11	6.82	6.75	8.06	8.78	7.1	6.75	7.53	7.46	7.05	6.14
37	8.06	8.52	7.53	6.66	8.72	8.44	7.46	6.96	8.82	7.86	6.68	6.02
38	8.06	8.18	7.36	6.28	8.06	8.5	7.34	6.28	8.4	7.66	5.53	5.99
39	8.3	8.26	7.05	6.48	8.18	8.96	7.1	6.39	8.85	7.91	6.7	5.74
40	8.09	8.53	7.66	6.65	8.66	8.47	6.99	6.28	8.17	8.81	5.89	6.43
41	8.7	8.17	7.59	6.66	8.9	8.18	7.03	6.75	9.17	9.11	5.73	5.44
42	8.29	8.28	6.71	6.59	8.46	8.82	7.34	6.75	9.07	7.46	6.49	6.12
43	8.2	8.22	7.69	6.75	9.2	7.83	7.22	6.75	9.21	7.96	7.22	6.05
44	8.07	9.16	7.36	7.05	8.36	8.92	7.14	6.75	8.57	7.94	6.08	5.82
45	8.72	8.62	6.92	6.37	8.06	8.51	7.35	6.44	8.22	7.28	5.88	5.89
46	9.05	7.98	7.75	6.44	7.97	7.93	7.5	6.75	7.61	8.88	6.42	5.96
47	8.57	8.76	7.18	6.39	8.24	8.8	7.28	6.56	8.41	9.13	6.09	5.9
48	8.38	8.47	7.28	6.44	8.62	8.68	7.82	7.34	8.11	8.99	5.34	5.59
49	8.09	8.22	7.03	6.75	7.68	8.21	7.34	6.37	8.28	8.74	6.22	5.75
50	8.1	8.29	7.66	6.38	8.44	8.17	7.4	6.38	8.48	9.77	5.69	5.97
Min	7.68	7.72	6.64	6.18	7.40	7.73	6.66	5.77	7.32	6.91	5.34	5.44
Max	9.48	9.37	7.82	7.13	9.59	9.21	7.83	7.34	9.41	9.77	7.30	6.72
Ave	8.45	8.43	7.20	6.55	8.36	8.42	7.17	6.52	8.21	8.13	6.26	6.02
Stdev	0.38	0.39	0.33	0.22	0.43	0.39	0.28	0.30	0.50	0.68	0.46	0.28

Appendix E: UMP semester1-200708 *modified*-GDA results (*cont...*)

No.	Nh9 – 3000 iterations				Nh10 – 1500 iterations				Nh10-3000 iterations			
	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82	16.68	13.74	10.30	7.82
1	8.89	7.21	5.87	5.36	8.82	8.84	7.41	6.98	8.52	8.87	7.09	6.65
2	8.05	7.8	5.76	6.04	8.46	8.44	7.44	6.92	8.89	8.43	7.38	5.94
3	8.2	7.74	5.71	6.17	9.1	8.97	6.79	6.79	8.53	8.02	6.88	6.28
4	7.16	8.47	5.9	5.95	8.85	9.3	7.45	6.73	7.97	8.15	7.45	7.07
5	8.76	8.35	6.24	5.45	8.24	8.83	7.07	6.72	8.6	8.28	7.42	6.92
6	7.22	8.86	5.46	6.74	8.99	8.62	6.86	6.92	8.39	9.02	7.78	6.34
7	8.39	7.88	6.19	6.78	9.27	9.08	7.4	6.92	8.51	8.02	7.05	6.92
8	7.9	7.9	5.24	5.84	9.1	8.55	7.62	6.38	9.13	8.05	7.02	6.17
9	7.42	7.79	6	6.34	8.37	8.27	7.29	5.93	8.12	8.32	7.28	6.92
10	8.17	7.91	6.4	5.94	7.82	8.21	7.82	7.03	8.8	8.56	7.19	6.17
11	8.03	7.82	6.59	6.34	9.3	8.63	7.65	6.92	9.3	9.15	6.97	6.17
12	9.36	7.54	6.28	5.87	8.81	8.74	7.24	6.92	8.61	8.35	7.06	6.07
13	7.64	8.19	5.42	6.46	9.06	8.13	7.34	6.92	9.03	9.08	7.13	6.92
14	7.65	7.55	5.87	6.25	9.15	8.89	7.89	6.38	9	9.44	8.32	6.43
15	7.75	6.9	6.61	6.13	9.51	8.34	7.48	6.28	8.12	8.95	6.88	6.17
16	8.58	7.37	6.54	6.74	9.18	9.4	7.16	6.92	9.22	8.6	7.57	6.92
17	7.96	6.96	5.54	6.3	8.28	8.09	7.52	5.93	8.6	9.13	6.94	6.65
18	7.68	7.05	5.31	5.91	8.85	8.01	7.75	6.72	8.41	7.88	7.44	6.92
19	7.31	7.67	6.26	6.05	8.73	8.44	7.55	6.25	8.92	8.23	6.57	6.81
20	7.55	7.41	5.44	5.82	8.88	8.59	7.07	7.07	8.98	8.05	6.57	6.17
21	7.64	8.24	5.23	5.62	8.55	8.35	7.34	6.28	8.47	8.1	7.77	7.03
22	7.78	8.7	5.61	5.35	7.97	8.14	7.33	6.26	8.18	8.81	7.43	6.48
23	7.82	7.8	6.69	5.98	9.08	8.11	7.05	5.93	8.49	8.35	7.45	6.92
24	8.37	7.71	5.74	6.11	9.28	9.02	8.2	7.03	8.21	8.54	7.27	6.15
25	8.28	7.59	7.06	6.69	8.41	8.15	7.6	6.92	9.61	8.51	7.11	6.92
26	8.45	7.26	5.9	6.28	9.1	8.68	7.57	6.92	9.33	8.59	6.99	6.85
27	8.37	7.87	6.12	6.1	8.55	9.31	7.04	6.92	8.52	8.62	7.78	6.17
28	7.92	7.71	5.19	5.54	8.37	8.1	7.67	6.92	8.53	8.12	7.2	6.17
29	7.79	8.82	5.44	6.06	8.21	8.89	7.94	6.16	8.68	8.94	7.77	6.63
30	8.31	7.84	6.22	6.09	7.79	8.19	7.38	6.26	7.88	8.68	7.07	6.78
31	7.44	7.01	7.08	5.68	8.49	9.01	7.49	6.17	7.82	8.68	7.07	6.28
32	8.45	8.18	5.59	6.47	8.78	8.1	7.16	7.07	8.59	8.13	8.12	6.17
33	7.81	7.99	4.92	6.17	9.86	8.35	7.41	6.92	8.93	8.46	7.34	6.92
34	8.44	7.92	5.79	5.86	8.84	8.76	7.97	6.92	8.12	8.58	7.07	6.98
35	8.05	7.84	5.68	5.49	8.62	8.31	7.16	6.92	8.66	8.1	7.36	6.65
36	7.61	7.86	6.45	6.04	8.27	8.14	7.45	6.92	9.48	8.22	7.46	6.63
37	7.96	8.41	7.67	5.59	9.88	8.63	7.79	6.17	8.31	8.49	7.28	6.01
38	8.85	7.62	6.07	5.98	8.27	8.62	7.46	7.03	8.62	8.07	7.47	7.03
39	8.21	7.37	7.45	5.98	9	8.71	7.56	6.07	8.41	8.93	7.12	5.92
40	7.95	7.32	6.91	6.46	8.15	8.71	8.34	6.92	8.5	8.62	7.14	6.01
41	8.11	7.03	5.79	5.59	8.26	8.65	7.68	5.83	7.97	8.53	7.05	6.98
42	8.66	7.08	5.9	6.13	9.1	8.52	7.62	6.92	8.27	7.95	7.18	6.17
43	8.06	8.02	5.8	6.14	8.64	8.23	7.87	6.92	8.78	8.23	7.35	6.92
44	7.88	7.45	6.51	6.04	8.78	8.08	7.76	6.13	8.47	8.89	7.29	6.97
45	8.59	7.83	5.23	6.1	9.32	8.26	6.94	6.19	8.43	8.52	7.46	6.92
46	8.39	8	6.77	6.03	8.95	8.71	7.57	6.98	8.08	8.75	7.29	6.14
47	8.11	7.35	6.02	6.42	8.85	8.39	7.33	6.8	8.58	8.44	7.33	6.72
48	7.95	7.66	5.95	5.7	8.95	8.58	7.31	6.24	9.13	7.99	6.85	6.92
49	8.46	7.91	6.47	6.04	8.75	8.6	7.23	6.98	9.4	8.79	8.44	5.93
50	7.96	8.31	6.05	6.03	8.61	8.74	7.76	6.92	7.97	8.01	7.76	6.92
Min	7.16	6.90	4.92	5.35	7.79	8.01	6.79	5.83	7.82	7.88	6.57	5.92
Max	9.36	8.86	7.67	6.78	9.88	9.40	8.34	7.07	9.61	9.44	8.44	7.07
Ave	8.07	7.76	6.04	6.04	8.77	8.55	7.48	6.65	8.60	8.48	7.31	6.56
Stdev	0.45	0.47	0.60	0.35	0.46	0.35	0.32	0.39	0.44	0.37	0.38	0.38

Appendix F: UMP semester1-200809 *modified*-GDA results

No.	Nh1- 1500 iterations				Nh1 – 3000 iterations				Nh2 – 1500 iterations			
	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21
1	8.46	7.2	6.74	6.27	7.53	7.54	6.25	5.98	18.33	15.25	12.3	9.21
2	8.24	7.59	6.59	6.5	7.66	6.75	6.67	5.89	18.33	15.25	12.3	9.21
3	7.82	7.28	6.74	6.34	7.6	7.57	6.3	5.93	18.34	15.25	12.3	9.21
4	8.65	7.55	6.83	6.16	7.52	6.91	6.57	6.2	18.34	15.25	12.3	9.21
5	8.34	7.52	6.58	6.17	8.03	7.36	6.39	6.1	18.34	15.25	12.3	9.21
6	8.06	7.38	6.64	6.37	7.94	6.72	6.73	6.11	18.33	15.25	12.3	9.21
7	8.68	7.62	6.72	6.6	7.26	6.98	6.03	6.05	18.34	15.25	12.3	9.21
8	8.42	8.05	7.05	6.73	7.74	6.7	6.49	6.03	18.34	15.25	12.3	9.21
9	8.33	7.76	6.65	6.3	7.62	6.29	6.51	6.13	18.34	15.25	12.3	9.21
10	7.86	6.89	7.03	6.78	7.71	7.22	6.58	5.98	18.33	15.25	12.3	9.21
11	8.58	7.67	6.36	6.35	7	7.26	6.89	6	18.34	15.25	12.3	9.21
12	8.21	7.71	6.94	6.6	7.42	6.82	6.37	6.14	18.34	15.25	12.3	9.21
13	8.92	7.72	6.89	6.35	7.13	7.05	6.23	5.92	18.33	15.25	12.3	9.21
14	8.63	7.57	6.8	6.4	7.52	7.13	6.38	5.98	18.34	15.25	12.3	9.21
15	9.37	7.41	6.92	6.59	7.36	6.92	6.32	6.21	18.34	15.25	12.3	9.21
16	8.91	7.89	6.54	6.52	7.65	6.72	6.18	6.2	18.34	15.25	12.3	9.21
17	8.35	7.84	6.69	6.58	7.27	7.22	6.41	5.98	18.34	15.25	12.3	9.21
18	8.13	7.37	6.84	6.79	7.45	6.86	6.86	6.34	18.34	15.25	12.3	9.21
19	8.09	7.84	6.73	6.19	8.26	7.14	6.3	6.01	18.34	15.25	12.3	9.21
20	8.17	7.65	7.24	6.72	7.74	6.71	6.46	6.2	18.34	15.25	12.3	9.21
21	8.42	7.27	6.82	6.41	7.03	6.97	6.24	6.11	18.33	15.25	12.3	9.21
22	8.58	7.74	6.33	6.42	7.86	7.55	6.41	6.03	18.34	15.25	12.3	9.21
23	8.57	7.74	7.16	6.55	7.4	7.24	6.63	6	18.34	15.25	12.3	9.21
24	8.76	7.48	6.83	6.28	7.58	6.68	6.18	6.09	18.34	15.25	12.3	9.21
25	8.02	7.58	7.13	6.55	7.47	7.04	6.26	5.9	18.34	15.25	12.3	9.21
26	7.61	7.49	7.02	6.2	7.62	7.14	6.19	5.89	18.33	15.25	12.3	9.21
27	8.2	7.58	6.95	6.24	7.94	7.06	6.34	5.97	18.33	15.25	12.3	9.21
28	8.49	7.44	6.92	6.3	7.22	7.02	6.41	5.95	18.34	15.25	12.3	9.21
29	8.3	7.71	6.66	6.58	7.72	6.78	6.26	6.32	18.33	15.25	12.3	9.21
30	8.63	7.92	6.83	6.11	7.79	6.97	6.25	6.02	18.33	15.25	12.3	9.21
31	8.84	7.22	6.94	6.78	7.45	6.91	6.25	6.04	18.34	15.25	12.3	9.21
32	8.14	8.01	7.16	6.35	7.71	7.23	6.66	6.11	18.33	15.25	12.3	9.21
33	8.43	8.14	6.88	6.23	7.47	7.74	6.19	5.82	18.33	15.25	12.3	9.21
34	7.87	7.56	6.92	6.73	7.44	7.05	6.46	5.99	18.33	15.25	12.3	9.21
35	8.15	7.26	6.76	6.28	7.52	7.01	6.59	5.95	18.33	15.25	12.3	9.21
36	8.32	7.13	6.77	6.37	7.52	7.55	6.34	6.42	18.33	15.25	12.3	9.21
37	8.59	8.11	7.22	6.33	7.96	7.16	6.38	6.13	18.32	15.25	12.3	9.21
38	8.62	7.28	6.84	6.87	7.51	7.04	6.23	6.04	18.34	15.25	12.3	9.21
39	8.76	8.14	7.05	6.52	7.41	6.97	6.46	6.29	18.34	15.25	12.3	9.21
40	8.54	7.26	6.92	6.56	7.24	6.71	6.52	6.15	18.33	15.25	12.3	9.21
41	8.81	8.04	7.05	6.28	7.46	7.5	6.35	5.9	18.34	15.25	12.3	9.21
42	8.72	7.6	7.61	6.44	7.95	6.83	6.05	5.83	18.34	15.25	12.3	9.21
43	8.48	7.38	7.44	6.78	8.05	7.29	6.37	5.63	18.34	15.25	12.3	9.21
44	8.57	8.18	6.82	6.14	7.19	7.14	6.36	6.15	18.33	15.25	12.3	9.21
45	8.2	7.64	6.86	6.4	7.77	7.47	6.06	5.89	18.34	15.25	12.3	9.21
46	8.84	8.01	7	6.37	6.96	6.85	6.58	5.88	18.33	15.25	12.3	9.21
47	8.39	7.41	6.85	6.29	7.35	6.94	6.23	6.03	18.34	15.25	12.3	9.21
48	8.24	7.97	6.79	6.39	7.65	7.7	6.31	6.14	18.33	15.25	12.3	9.21
49	8.05	7.11	7.16	6.54	7.36	6.96	6.03	5.8	18.34	15.25	12.3	9.21
50	8.8	7.52	7.2	6.25	7.66	6.99	6.72	6.06	18.34	15.25	12.3	9.21
Min	7.61	6.89	6.33	6.11	6.96	6.29	6.03	5.63	18.32	15.25	12.30	9.21
Max	9.37	8.18	7.61	6.87	8.26	7.74	6.89	6.42	18.34	15.25	12.30	9.21
Ave	8.42	7.61	6.89	6.44	7.55	7.07	6.38	6.04	18.34	15.25	12.30	9.21
Stdev	0.33	0.31	0.24	0.20	0.28	0.30	0.20	0.15	0.01	0.00	0.00	0.00

Appendix F: UMP semester1-200809 *modified*-GDA results (*cont...*)

No.	Nh2 – 3000 iterations				Nh3- 1500 iterations				Nh3 – 3000 iterations			
	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21
1	18.33	15.25	12.3	9.21	7.89	9.4	7.67	7.74	7.01	8.21	7.57	7.2
2	18.34	15.25	12.3	9.21	8.43	8.34	8.36	7.67	7.43	7.9	8.36	7.28
3	18.34	15.25	12.3	9.21	7.68	8.44	8.05	7.57	6.86	7.83	7.75	6.65
4	18.33	15.25	12.3	9.21	7.83	8.7	8.22	7.46	7.15	8.63	8.07	7.23
5	18.34	15.25	12.3	9.21	7.91	8.98	8.67	7.44	7.43	8.28	8.34	6.77
6	18.34	15.25	12.3	9.21	8.54	8.95	8.23	7.67	7.75	8.12	8.09	7.59
7	18.34	15.25	12.3	9.21	8.69	8.63	8.69	7.34	7.27	8.42	8.07	7.12
8	18.34	15.25	12.3	9.21	7.69	9.2	8.07	7.22	7.04	8.24	8.12	6.79
9	18.33	15.25	12.3	9.21	7.87	8.86	8.24	7.72	6.96	8.25	8.11	7.04
10	18.34	15.25	12.3	9.21	8.19	8.6	8.78	7.36	7.83	7.87	7.4	6.38
11	18.34	15.25	12.3	9.21	7.88	8.55	8.11	7.35	8.03	7.76	8.36	7.4
12	18.33	15.25	12.3	9.21	7.88	9.11	7.99	7.63	7.48	8.56	7.66	7.64
13	18.34	15.25	12.3	9.21	7.68	8.07	7.97	7.17	7.61	7.87	7.63	7.29
14	18.34	15.25	12.3	9.21	7.7	9.32	7.72	7.97	7.05	7.92	8.31	7.56
15	18.33	15.25	12.3	9.21	7.67	8.61	7.85	7.32	7.04	8.52	8.24	6.82
16	18.34	15.25	12.3	9.21	8.45	8.9	8.49	7.96	6.78	8.11	7.77	7.23
17	18.34	15.25	12.3	9.21	8.21	9.11	7.66	7.4	7.29	8.51	8.38	6.84
18	18.33	15.25	12.3	9.21	7.79	8.85	8.9	7.6	7.17	8.55	7.9	7.62
19	18.33	15.25	12.3	9.21	7.85	9.09	8.49	7.31	6.95	7.97	7.6	7.26
20	18.34	15.25	12.3	9.21	7.65	8.44	8.14	7.09	7.42	8.36	8.22	6.81
21	18.34	15.25	12.3	9.21	7.87	8.83	7.99	7.73	6.92	8.32	8.26	6.9
22	18.34	15.25	12.3	9.21	7.72	8.84	7.94	7.6	7.61	7.7	8.35	6.99
23	18.34	15.25	12.3	9.21	8.3	8.66	8.38	7.41	7.12	8.54	7.65	7.1
24	18.33	15.25	12.3	9.21	7.86	8.79	7.87	7.69	7.05	8.03	7.75	7.05
25	18.34	15.25	12.3	9.21	7.12	9.25	8.43	7.56	7.43	8.57	8.01	6.85
26	18.33	15.25	12.3	9.21	8.66	8.66	7.84	7.85	7.12	8.48	7.32	6.83
27	18.34	15.25	12.3	9.21	8.14	8.6	7.91	7.85	7.02	8.56	7.54	7.42
28	18.34	15.25	12.3	9.21	7.71	8.85	8.63	7.76	7.88	8.3	8.1	7.27
29	18.33	15.25	12.3	9.21	8.02	8.62	7.98	7.37	6.79	8.61	7.58	6.99
30	18.34	15.25	12.3	9.21	8.02	8.74	7.88	7.38	7.17	7.71	8.07	7.35
31	18.33	15.25	12.3	9.21	8.6	8.58	8.01	7.74	6.83	8.37	7.85	7.56
32	18.34	15.25	12.3	9.21	7.71	8.55	8.3	7.54	7.4	8.11	8.8	6.83
33	18.34	15.25	12.3	9.21	8.06	8.81	8.83	7.86	7.85	8.27	8.54	6.66
34	18.34	15.25	12.3	9.21	8.6	8.73	8.43	6.95	7.28	7.8	7.75	6.82
35	18.33	15.25	12.3	9.21	8.05	8.71	8.5	7.42	6.98	8.31	8.03	7.21
36	18.33	15.25	12.3	9.21	8.47	9.08	8.93	7.56	7.25	8.13	8.04	7.26
37	18.33	15.25	12.3	9.21	7.37	8.66	8.35	7.55	7.33	8.39	8	7.01
38	18.34	15.25	12.3	9.21	7.75	8.68	8.25	7.82	6.92	8.07	8.04	7.11
39	18.34	15.25	12.3	9.21	7.55	9.54	8.51	7.61	7.06	8.26	8.09	7.61
40	18.34	15.25	12.3	9.21	8.47	9.85	8.1	7.17	7.26	8.04	7.73	7.22
41	18.33	15.25	12.3	9.21	7.9	8.73	8.27	8.25	7.35	8.64	8.1	7.38
42	18.33	15.25	12.3	9.21	8.15	8.68	8.36	7.64	7.17	7.86	8.26	7.29
43	18.34	15.25	12.3	9.21	8.09	9.04	8.99	7.88	7.62	8.71	7.8	6.93
44	18.33	15.25	12.3	9.21	7.8	8.46	8.1	7.9	7.43	8.43	7.71	6.79
45	18.33	15.25	12.3	9.21	7.71	8.71	8.35	7.6	7.53	8.03	8.38	7.82
46	18.33	15.25	12.3	9.21	8.68	8.3	7.69	7.45	7.34	8.33	8.3	6.77
47	18.33	15.25	12.3	9.21	8.18	8.95	8.01	7.51	7.08	8.06	8.15	7.01
48	18.34	15.25	12.3	9.21	7.84	8.75	8.28	7.17	7.09	8.39	7.96	7.33
49	18.33	15.25	12.3	9.21	7.66	9.18	8.86	7.44	7.25	8.09	8.75	7.15
50	18.34	15.25	12.3	9.21	8.28	8.5	8.36	7.47	7.3	8.48	8.09	7.55
Min	18.33	15.25	12.30	9.21	7.12	8.07	7.66	6.95	6.78	7.70	7.32	6.38
Max	18.34	15.25	12.30	9.21	8.69	9.85	8.99	8.25	8.03	8.71	8.80	7.82
Ave	18.34	15.25	12.30	9.21	8.00	8.81	8.25	7.55	7.26	8.23	8.02	7.13
Stdev	0.00	0.00	0.00	0.00	0.36	0.33	0.35	0.25	0.30	0.28	0.33	0.31

Appendix F: UMP semester1-200809 *modified*-GDA results (*cont...*)

No.	Nh4 – 1500 iterations				Nh4 – 3000 iterations				Nh6-1500 iterations			
	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21
1	9.95	9.13	8.38	7.46	10.29	10.06	8.33	7.65	9.55	11.02	10.29	9.03
2	10.11	9.26	8.41	7.44	10.04	8.93	8.65	7.36	11.63	11.42	9.29	9.03
3	10.24	9.48	8.34	7.73	11.06	9.49	8.24	7.6	11.27	11.23	9.67	9.03
4	9.67	9.12	8.22	7.26	9.46	9.74	9.55	7.31	11.92	10.3	9.81	8.99
5	10.18	8.79	7.66	7.37	10.39	9.05	8.3	7.28	10.09	11.3	9.49	8.99
6	10.07	9.47	8.15	7.34	9.28	9.59	8.41	7.22	12.12	11.31	9.4	9.03
7	9.77	9.93	8.52	7.27	9.61	8.92	7.77	7.09	10.34	10.71	9.9	8.95
8	10.48	9.65	9.18	7.27	10.61	9.51	8.35	7.44	9.72	10.82	10.3	9.14
9	9.94	9.53	9.09	7.46	9.35	9.37	8.36	7.28	10.47	10.98	9.85	9.03
10	10.4	9.76	8.07	7.26	9.63	8.89	8.46	7.85	10.08	11.08	9.88	8.96
11	9.86	9.2	9.09	7.45	9.34	9.31	8.29	7.69	10.29	10.81	9.3	8.68
12	10.46	10	8.12	7.17	9.98	8.62	8.12	7.31	10.2	11.19	10.3	9.03
13	9.72	8.58	8.49	7.52	9.81	8.72	8.49	7.49	9.81	10.65	10.15	8.96
14	10.21	9.3	8.72	7.5	9.48	9.25	8.03	7.31	11.42	11.28	9.82	9.14
15	11.1	8.79	8.21	7.73	9.09	9.89	9.29	7.71	11.58	10.94	9.78	8.15
16	10.06	9.54	8.35	7.66	9.48	8.67	8.87	7.36	11.37	11.21	9.85	8.68
17	10.09	9.65	8.77	7.63	9.8	9.1	8.39	7.53	11.42	10.98	9.34	8.95
18	9.88	9.47	8.49	7.49	9.15	9.67	8.42	7.37	14.41	11.61	10.1	9.03
19	10.95	9.58	8.39	7.14	9.96	9.41	8.36	7.5	11.53	10.99	9.88	8.95
20	9.96	9.07	8.2	7.63	10.06	9.23	8.04	7.31	11.52	10.85	9.52	8.15
21	10.27	9.53	8.92	7.66	9.34	9.39	8.28	7.66	11.35	11.19	9.6	9.03
22	9.88	9.01	8.18	7.68	9.81	9.58	7.97	7.46	9.86	11.03	9.82	8.96
23	10.38	9.86	8.59	7.38	10.13	9.07	7.72	7.6	9.88	10.67	10.15	9.03
24	10.16	9.04	8.41	7.43	10.21	8.78	8.03	7.41	11.88	11.31	9.84	9.03
25	10.72	8.75	8.12	7.52	10.19	8.68	8.66	7.43	11.36	11	9.56	9.03
26	10.13	8.74	8.39	7.29	9.3	8.45	8.35	6.97	11.45	11.1	9.59	9.03
27	10.97	9.54	8.78	7.32	10.27	8.75	7.94	7.39	10.88	10.82	9.93	9.03
28	9.6	10.25	8.86	7.42	9.27	9.83	8.17	7.78	9.93	11.09	10	9.03
29	9.71	9.81	8.75	7.62	9.57	9.48	8.65	7.49	9.91	10.86	9.39	9.03
30	9.75	9.2	8.71	7.61	10.51	9.5	8.53	7.39	10.07	10.75	9.96	9.14
31	9.75	9.63	8.02	7.83	9.8	9.31	8.45	7.82	14.57	10.68	9.56	9.14
32	9.11	9.51	8.16	7.4	9.79	9.89	8.71	7.68	11.55	11.17	10.09	9.03
33	10.57	10	8.39	7.38	10.94	9.15	8.84	7.68	11.56	10.88	9.34	8.96
34	9.92	9.35	8.31	7.56	9.63	10.03	7.94	7.62	11.28	10.74	9.55	9.03
35	11.41	9.08	8.61	7.8	9.76	9.1	8.81	7.65	9.15	10.94	9.8	8.68
36	10.51	9.3	8.13	7.51	9.98	9.47	8.48	7.62	9.3	11	10.12	9.03
37	9.92	9.23	8.48	7.61	9.84	9.71	8.39	7.22	11.9	10.64	10.14	9.03
38	9.57	9.45	8.53	7.57	10.66	8.99	8.49	7.54	11.11	11.05	9.84	9.03
39	9.65	8.95	8.98	7.27	10.34	9.79	8.31	7.34	11.54	10.99	9.99	8.96
40	10.1	9.95	8.41	7.49	10.49	9.6	8.37	7.66	11.39	11.18	10	8.68
41	10.93	9.64	8.28	7.39	9.45	9.58	7.87	7.67	10.35	11.07	9.55	9.03
42	11.03	9.91	8.28	7.26	10.07	7.92	9.04	7.55	10.02	10.72	10.1	9.03
43	9.14	8.52	8.03	7.43	11.22	9.98	7.78	7.71	10.74	11.02	10.06	9.03
44	10.12	9.79	8.61	7.26	9.42	9.39	8.47	7.7	10.89	10.63	9.57	8.96
45	9.92	9.43	8.45	7.84	9.74	8.86	8.12	7.54	12.1	10.85	9.61	9.03
46	9.69	8.66	8.42	7.24	9.19	9.75	8.22	7.6	11.45	10.92	10.54	8.96
47	10.79	10.02	8.49	7.53	10.7	8.94	8.2	7.62	11.2	11.08	9.67	9.03
48	10.39	9.55	8.44	7.33	9.58	8.93	8.12	7.16	10.1	10.7	9.59	9.03
49	11.14	9.4	8.73	7.75	10.4	8.76	8.33	7.3	11.52	11.22	9.61	9.03
50	9.33	9.6	8.81	7.58	9.98	9.92	9.04	7.27	9.73	10.85	9.47	9.03
Min	9.11	8.52	7.66	7.14	9.09	7.92	7.72	6.97	9.15	10.30	9.29	8.15
Max	11.41	10.25	9.18	7.84	11.22	10.06	9.55	7.85	14.57	11.61	10.54	9.14
Ave	10.15	9.40	8.46	7.47	9.91	9.28	8.38	7.48	10.98	10.98	9.80	8.96
Stdev	0.52	0.41	0.31	0.18	0.52	0.47	0.37	0.20	1.09	0.24	0.30	0.20

Appendix F: UMP semester1-200809 *modified*-GDA results (*cont...*)

No.	Nh6 – 3000 iterations				Nh7 – 1500 iterations				Nh7-3000 iterations			
	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21
1	11.42	11.46	10.34	9.03	9.85	8.08	7.24	6.74	8.58	7.58	7.08	5.86
2	10.11	11.51	9.76	9.03	9.69	8.05	6.91	7.01	8.8	7.9	6.94	6.64
3	9.61	11	9.71	9.14	8.58	7.57	6.93	6.73	8.71	7.48	7.01	6.29
4	11.66	11.12	10.13	9.14	9.77	8.06	7.01	6.36	8.49	8.22	6.63	6.34
5	10.23	11.15	10.15	9.03	8.82	8.07	7.26	6.44	8	7.4	6.29	6.72
6	10.18	10.81	9.54	9.03	9.21	8.31	7	6.77	8.3	7.73	6.62	6.47
7	9.27	11	9.72	9.03	9.23	7.89	7.22	6.43	8.57	8.42	6.88	6.65
8	9.73	10.92	9.65	9.03	8.53	7.77	8.07	6.85	8.48	7.19	6.7	6.42
9	11.16	10.75	9.83	8.68	9.2	7.93	7.01	6.74	8.77	8.2	6.5	6.85
10	10.09	11.17	9.35	9.03	8.79	8.23	7.02	6.72	8.17	7.43	6.68	6.08
11	9.5	10.99	9.98	9.02	8.42	8.31	6.75	7.12	8.43	7.69	6.9	6.38
12	10.46	11.14	10.15	9.03	8.57	7.94	6.7	6.5	8.48	7.58	6.63	6.72
13	9.88	11.23	9.63	9.03	8.89	8.54	6.79	6.33	8.71	7.91	6.8	6.57
14	9.86	11.04	10.03	9.21	8.78	8.35	7.08	6.31	9.01	7.4	6.77	6.41
15	9.21	10.76	9.42	9.03	9.52	7.91	6.8	6.58	8.6	7.9	6.83	6.2
16	9.13	11.14	9.65	9.21	8.61	8.13	6.62	7.09	8.57	7.23	6.45	6.21
17	11.67	12	10.03	9.03	8.74	8.48	7.05	6.79	8.73	8.01	6.64	6.17
18	11.72	10.98	10.17	9.03	9.5	8.81	6.97	6.37	9.06	8.04	6.59	6.05
19	11.38	11.35	9.58	9.03	8.63	7.8	7.5	6.38	8.62	7.63	7.03	6.41
20	10.96	11.3	10.02	9.03	8.83	8.55	7.23	6.87	8.23	8	6.91	6.57
21	11.46	11.22	9.8	9.03	9.42	9.12	6.75	6.94	8.31	8.33	6.73	6.28
22	10.34	11.09	9.87	9.03	9.2	8.51	6.81	6.86	8.47	8.08	6.67	6.29
23	9.66	10.96	9.39	9.03	8.62	8.1	7.01	6.96	8.68	8.27	6.28	6.3
24	9.39	11.02	9.38	8.59	8.86	8.06	6.8	6.52	7.99	7.3	6.26	6.17
25	10.96	10.83	10.24	9.03	9.13	8.18	7.19	6.39	9.28	7.6	7.4	6.28
26	10.14	10.34	9.61	9.03	8.88	9.2	6.92	6.21	8.8	7.85	6.86	6.18
27	9.96	10.84	9.36	8.6	9.21	8.28	7.59	6.97	8.09	7.94	7.14	6.66
28	9.34	10.88	9.91	8.68	9.21	8.56	6.84	6.81	8.69	8.09	7.19	6.45
29	10.68	11.81	9.29	9.14	9.09	8.31	6.89	6.89	8.42	7.72	6.89	6.06
30	9.93	10.8	9.88	9.03	8.81	8.02	6.96	6.86	8.27	7.97	6.87	6.75
31	11.25	11.36	10.18	9.03	9.41	8.34	6.78	6.39	8.07	9.28	6.95	6.66
32	10.02	10.87	10.25	9.03	9	9.27	6.77	6.68	8.91	8.13	6.34	6.59
33	10.23	10.56	9.56	9.21	8.79	7.77	6.98	7.35	8.24	7.72	7	6.82
34	10.08	11.07	10.1	9.14	9.19	7.99	7.35	6.74	9.3	7.53	6.75	6.17
35	10.8	11.07	9.86	8.95	8.88	8.26	6.89	6.44	8.19	7.76	6.48	6.26
36	9.49	10.99	9.84	8.96	10.01	8.93	7.01	6.9	8.48	7.56	6.84	6.19
37	10.87	10.91	10.03	9.03	8.71	7.9	7.1	6.47	8.26	7.72	6.69	6.45
38	11.36	10.99	9.76	9.03	9.1	7.8	6.77	6.93	8.25	7.55	6.99	6.3
39	11.23	10.6	9.21	7.86	8.61	7.96	6.98	6.87	8.43	7.97	6.64	6.63
40	10.06	11.26	9.4	9.21	8.62	7.81	6.79	6.8	8.57	7.6	6.55	6.56
41	9.84	11.29	9.48	9.21	8.64	8.34	7.13	7.17	8.5	7.39	6.44	6.02
42	9.42	10.92	10.15	9.03	9.05	8.75	6.95	6.72	8.33	8.56	6.19	6.64
43	10.46	11	10.41	9.03	9.06	8.15	7.18	7.01	8.52	7.82	6.75	6.35
44	8.98	11.52	10.1	8.58	8.85	8.2	7.12	6.79	8.67	6.96	6.65	6.27
45	9.6	11.06	9.84	8.99	8.45	7.74	7.08	6.69	8.41	7.4	6.86	5.95
46	9.42	10.73	9.57	9	8.34	7.67	7.57	6.3	8.78	7.54	6.92	6.41
47	9.8	11.12	9.86	9.21	9.37	8.17	6.77	6.53	8.31	7.98	6.69	6.14
48	10.38	11.02	9.53	9.1	8.62	8.26	7.03	6.83	8.47	7.92	6.39	6.13
49	10.06	11.48	9.42	9.03	9.46	8.33	7.08	6.37	8.78	7.37	6.82	6.3
50	9.75	10.98	9.87	9.21	8.81	7.81	7.12	7.07	8.62	7.54	6.58	6.27
Min	8.98	10.34	9.21	7.86	8.34	7.57	6.62	6.21	7.99	6.96	6.19	5.86
Max	11.72	12.00	10.41	9.21	10.01	9.27	8.07	7.35	9.30	9.28	7.40	6.85
Ave	10.24	11.07	9.80	9.00	8.99	8.21	7.03	6.71	8.53	7.79	6.73	6.37
Stdev	0.76	0.30	0.31	0.22	0.39	0.39	0.26	0.27	0.29	0.40	0.25	0.24

Appendix F: UMP semester1-200809 *modified*-GDA results (*cont...*)

No.	Nh8 - 1500 iterations				Nh8 - 3000 iterations				Nh9-1500 iterations			
	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21
1	10.61	9.84	9.54	8.6	10.16	9.71	9.22	8.6	11.47	10.24	8.38	7.55
2	10.14	10.34	9.52	8.48	9.31	9.49	9.83	8.57	11.08	9.85	8.41	7.6
3	11.03	9.9	9.93	8.57	10.29	10.56	9.53	8.48	10.98	10.34	8.07	7.61
4	11.47	10.45	9.81	8.56	9.32	10.85	9.82	8.6	9.8	9.46	8.22	7.32
5	10.43	10.49	10.03	8.6	10.93	9.04	9.64	8.56	11.68	9.78	8.83	8.02
6	10.61	10.42	9.74	8.6	10.3	10.17	9.33	8.58	9.97	10.6	8.79	7.78
7	11.12	10.22	9.47	8.6	9.94	9.25	9.46	8.58	11.45	9.75	9.05	7.51
8	10.64	10.78	9.44	8.6	9.88	10.16	9.58	8.6	10.51	9.99	8.84	7.42
9	10.89	9.86	10.31	8.6	10.23	9.52	10.12	8.6	11.53	9.79	8.48	7.41
10	10.93	9.87	9.84	8.56	10.41	10.14	10.03	8.6	11.07	9.91	8.47	7.71
11	10.9	10.63	9.3	8.6	10.2	9.93	9.93	8.68	11.82	9.51	8.19	7.39
12	10.61	10.39	9.1	8.6	10.65	9.88	10.16	8.56	11.5	10.18	8.47	7.7
13	10.95	10.15	9.6	8.6	9.99	9.36	9.35	8.56	11.08	10.29	8.84	7.53
14	10.37	9.6	9.7	8.56	10.57	9.46	9.88	8.56	10.78	9.85	8.75	7.88
15	10.55	11.04	8.96	8.6	9.98	9.14	9.64	8.58	10.79	10.58	8.68	7.26
16	9.86	10.37	9.25	8.6	10.31	9.69	9.76	8.56	10.71	9.19	8.14	7.53
17	11.43	11.1	9.94	8.6	11.05	9.48	9.52	8.48	10.79	9.58	8.17	7.36
18	10.69	9.86	9.4	8.6	10.3	9.85	10.29	8.58	11.74	9.87	8	7.58
19	10.8	10.62	9.25	8.48	10.19	10.75	9.26	8.6	10.91	9.61	8.75	7.18
20	10.92	10.32	9.91	8.6	10.77	9.74	9.92	8.6	11.62	9.56	8.33	7.49
21	11.85	10.35	10.16	8.48	10.68	9.48	9.69	8.76	11.41	10.47	8.69	7.93
22	11.16	10.85	9.82	8.56	10.27	9.52	10.36	8.6	10.6	9.72	8.64	7.64
23	11.07	11.07	9.64	8.56	10.1	10.47	9.24	8.48	10.64	9.55	8.65	7.05
24	10.28	10.83	9.81	8.56	9.86	9.56	9.68	8.57	10.86	8.97	8.54	7.31
25	10.9	9.86	9.48	8.6	10.65	9.31	10	8.48	11.12	9.96	8	7.47
26	10.99	10.56	10.24	8.57	10.04	9.32	9.63	8.56	10.72	9.23	8.13	7.53
27	10.97	11.27	8.96	8.48	10.31	10.43	9.1	8.6	10.43	9.48	8.13	7.51
28	11.09	9.92	9.45	8.57	10.18	9.54	10.12	8.56	11.77	9.75	8.11	7.86
29	11.07	10.85	10.09	8.69	11.07	9.58	9.06	8.72	10.67	10.06	8.61	7.88
30	11.09	10.51	9.94	8.48	9.74	9.42	10.06	8.56	10.93	10.2	8.23	7.43
31	10.57	10.42	9.55	8.48	10.42	10.37	10	8.56	9.79	10.64	8.57	7.05
32	10.92	10.33	9.63	8.66	9.54	9.61	10	8.6	10.44	9.77	8.44	7.54
33	10.55	9.78	9.84	8.6	10.02	9.02	9.65	8.6	10.28	10.21	8.3	7.75
34	10.28	10.58	10.02	8.6	9.97	10.28	9.64	8.6	11.19	9.97	8.85	7.64
35	11.35	10.28	10.06	8.48	10.23	9.18	9.6	8.6	11.17	10.2	8.18	7.34
36	10.79	10.22	9.74	8.56	10.1	9.41	9.95	8.56	11.6	10.15	8.46	7.45
37	10.89	10.55	9.58	8.6	10.12	9.84	9.47	8.56	11.32	10.18	8.74	7.89
38	10.27	9.05	9.36	8.6	10.53	9.73	9.85	8.56	10.59	10.03	8.26	7.45
39	10.45	10.81	9.57	8.6	10.81	10.88	9.64	8.57	10.55	9.94	7.82	7.22
40	11.9	10.71	9.91	8.64	10.75	9.16	9.77	8.48	10.6	9.91	8.6	7.72
41	10.49	10.4	9.14	8.56	10.87	10.17	9.75	8.68	11.16	9.68	9.15	7.5
42	10.55	10.21	10.22	8.48	10.17	10.3	9.65	8.56	10.57	10.1	8.34	7.81
43	10.62	10.55	9.64	8.6	9.81	9.29	9.06	8.48	11.28	9.85	8.48	7.36
44	10.34	10.56	9.3	8.48	10.11	10.18	9.4	8.56	11.34	9.17	9.14	7.38
45	11	10.38	9.31	8.48	9.49	9.45	10	8.6	11.27	10.25	8.25	7.54
46	10.91	10.37	9.91	8.75	11.12	10.14	9.74	8.48	10.7	9.6	8.84	7.8
47	11.16	10.15	9.45	8.6	9.52	10.64	9.5	8.57	10.56	9.66	8.43	7.27
48	10.75	11.09	9.44	8.48	9.78	9.97	8.98	8.6	10.24	9.44	8.84	7.02
49	10.65	10.48	9.66	8.48	9.92	9.94	10.27	8.56	11.93	9.55	8.21	7.03
50	9.39	9.86	10.24	8.6	10.78	9.08	9.19	8.6	10.86	10.33	8.84	7.59
Min	9.39	9.05	8.96	8.48	9.31	9.02	8.98	8.48	9.79	8.97	7.82	7.02
Max	11.90	11.27	10.31	8.75	11.12	10.88	10.36	8.76	11.93	10.64	9.15	8.02
Ave	10.79	10.38	9.66	8.57	10.23	9.79	9.69	8.58	10.96	9.88	8.49	7.52
Stdev	0.45	0.43	0.34	0.06	0.45	0.50	0.34	0.06	0.52	0.38	0.31	0.24

Appendix F: UMP semester1-200809 *modified*-GDA results (*cont...*)

No.	Nh9 - 3000 iterations				Nh10 - 1500 iterations				Nh10-3000 iterations			
	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21	18.4	15.25	12.30	9.21
1	10.76	9.21	8.5	6.84	10.24	10.44	9.5	8.43	10.62	9.7	9.69	8.43
2	10.84	10.36	8.41	7.68	10.48	9.29	9.47	8.57	10.46	10.99	9.56	8.43
3	11.05	9.8	8.09	7.46	9.51	9.3	9.57	8.43	10.73	10.67	10.21	8.43
4	11.5	9.65	8.28	7.65	10.84	9.9	9.83	8.58	9.52	9.74	9.89	8.57
5	10.73	10.33	9.38	7.57	11.31	10.37	9.57	8.73	11.17	10.46	9.57	8.61
6	10.72	9.86	8.29	7.67	10.41	9.95	9.56	8.43	9.67	10.21	9.38	8.43
7	10.39	10.53	7.84	7.79	10.69	10.45	9.6	8.6	9.93	9.56	9.86	8.57
8	10.36	9.68	8.66	7.14	11.15	9.67	9.83	8.43	9.6	10.18	9.91	8.57
9	10.79	10.46	7.67	7.72	10.08	10.6	9.57	8.3	10.62	10.22	9.63	8.43
10	11.84	9.77	8.07	7.3	10.76	10.12	11.15	8.43	10.52	10.05	9.6	8.43
11	10.99	9.73	8.26	7.58	10.8	9.77	10.01	8.43	10.23	10.27	9.48	8.43
12	10.46	9.5	8.9	7.21	10.16	10.45	9.81	8.69	9.93	9.84	9.56	8.43
13	10.69	9.5	8.81	7.63	10.23	11.41	9.61	8.56	11.04	9.57	9.75	8.43
14	10.1	9.57	8.59	7.53	10.03	10.17	9.89	8.3	10.4	10.15	9.62	8.43
15	11.99	9.98	8.25	7.43	10.9	10.6	9.67	8.43	10.57	10.45	9.45	8.57
16	9.99	10.39	8.42	7.63	10.4	10.2	9.6	8.57	10.3	9.72	9.63	8.3
17	10.57	9.81	8.24	7.74	10.53	10.7	10.08	8.43	10.34	10.05	9.57	8.57
18	11.02	9.8	8.52	7.3	10.63	9.91	9.61	8.73	10.08	9.77	9.59	8.43
19	10.84	10.14	8.96	7.38	10.23	9.97	9.4	8.43	10.11	9.81	9.83	8.57
20	11.27	9.88	8.74	7.48	10.63	9.25	9.56	8.43	10.51	10.94	9.6	8.43
21	10.04	9.75	8.38	7.2	11.33	10.77	9.91	8.68	9.89	10.59	9.9	8.43
22	11.15	9.72	8.57	7.65	10.68	9.76	9.56	8.56	10.02	9.57	9.47	8.3
23	9.87	9.99	7.8	7.19	10.22	9.91	9.45	8.43	10.77	9.93	9.77	8.43
24	9.93	9.67	8.94	7.4	11.53	10.9	9.56	8.43	11.1	10.92	9.57	8.43
25	10.78	9.87	8.3	7.6	10.86	10.5	9.56	8.52	9.94	10.25	9.59	8.43
26	11.59	10.25	8.29	7.18	10.79	9.89	10.29	8.43	11.03	9.21	9.56	8.43
27	10.82	10.48	8.6	7.08	10.19	9.99	9.57	8.43	10.18	9.55	9.5	8.43
28	11.46	10	8.05	7.41	10.34	10.88	10.45	8.43	10.44	9.26	10.21	8.37
29	11.53	9.09	9	7.87	10.34	10.2	9.8	8.43	10.52	9.12	9.79	8.43
30	10.75	10.25	8.26	7.36	10.23	10.43	9.44	8.6	10.01	9.76	9.93	8.56
31	10.57	9.39	7.91	7.22	11.43	10.2	9.91	8.57	9.82	10.51	9.47	8.56
32	11.27	9.09	7.72	7.48	11.09	9.63	9.37	8.43	10.42	9.82	9.66	8.43
33	10.96	9.74	8.31	7.43	11.67	10.15	9.57	8.43	10.16	10.26	9.71	8.43
34	10.97	9.64	8.38	7.61	10.65	9.78	9.87	8.43	10.38	10.36	9.56	8.43
35	10.48	10.27	8.34	7.3	11.19	10.04	10.21	8.7	10.97	9.83	9.89	8.43
36	10.76	9.66	8.69	7.52	10.92	9.56	9.64	8.6	10.45	10.01	9.48	8.57
37	11.11	9.76	8.29	7.56	10.71	10.21	9.99	8.43	11.14	8.96	9.56	8.57
38	10.47	9.94	8.93	7.58	9.77	10.31	9.81	8.39	10.59	9.63	9.48	8.43
39	9.83	9.65	8.61	7.24	10.57	10.35	9.63	8.56	9.69	9.86	9.56	8.76
40	10.72	9.74	8.2	6.78	10.5	10.57	9.81	8.43	11.84	10.94	9.66	8.61
41	9.72	9.63	8.13	7.51	10.72	9.94	9.57	8.43	10.82	10.45	9.6	8.43
42	10.15	10	8.52	7.51	10.62	11.18	9.81	8.3	10.9	10.03	9.71	8.57
43	10.67	9.71	8.85	7.68	10.54	10.07	9.48	8.43	10.47	9.9	10.16	8.57
44	10.68	10.47	8.18	7.5	10.55	9.62	9.91	8.43	10.19	10.34	9.87	8.43
45	10.65	9.3	7.98	7.71	10.02	10.22	9.39	8.64	10.2	10.22	9.43	8.43
46	10.62	10.34	7.97	7.6	10.38	11.02	9.59	8.43	10.11	9.83	9.7	8.57
47	9.77	10.06	8.34	7.44	10.99	10.51	10.02	8.43	10.51	10.2	9.73	8.43
48	10.92	10.2	8.55	7.06	10.04	10.73	9.67	8.43	10.09	10.95	9.4	8.43
49	10.95	9.21	8.34	7.71	10.34	11.03	9.65	8.57	10.02	9.73	9.78	8.43
50	10.69	9.9	8.2	7.49	10.46	10.62	9.26	8.43	10.8	10.63	9.44	8.57
Min	9.72	9.09	7.67	6.78	9.51	9.25	9.26	8.30	9.52	8.96	9.38	8.30
Max	11.99	10.53	9.38	7.87	11.67	11.41	11.15	8.73	11.84	10.99	10.21	8.76
Ave	10.74	9.85	8.39	7.45	10.59	10.23	9.73	8.49	10.40	10.06	9.67	8.48
Stdev	0.52	0.37	0.36	0.23	0.45	0.49	0.32	0.11	0.47	0.49	0.20	0.09

Appendix G: UMP semester1-200708 multi neighbourhood GDA results based on random ordering

No.	All Nh	Remove specified Nh						
		Nh5	Nh2	Nh10	Nh5&Nh2	Nh5&Nh10	Nh2&Nh10	Nh5, Nh2&Nh10
1	3.54	3.57	3.95	4.07	4.53	4.03	3.94	4.22
2	4.14	4.06	4.23	3.77	3.96	3.93	4.29	3.94
3	4.3	3.83	3.67	4.08	4.03	4.27	3.75	3.96
4	4.1	3.86	3.92	4.13	4.39	4.17	3.93	3.85
5	3.94	4.12	3.94	3.68	4.27	4.13	3.77	3.84
6	4	3.79	3.81	3.66	4.31	3.92	3.83	3.88
7	4.38	3.87	3.92	4	4.19	3.71	4.06	4.12
8	4.02	4.25	3.73	3.72	3.91	3.98	3.85	3.97
9	4.16	4.16	4.08	4.02	3.88	3.52	3.59	4.16
10	4.17	3.91	3.76	3.82	4.49	3.68	3.82	3.58
11	3.98	4.08	3.99	3.95	4.13	4.05	3.77	4.31
12	3.91	3.7	3.3	3.54	4.23	4.22	3.92	3.98
13	4.19	4.01	4.01	3.79	4.14	3.56	3.54	4.03
14	3.97	3.89	3.66	3.92	3.73	3.81	3.87	4.07
15	3.85	4.06	4.2	3.5	3.73	3.99	3.51	3.9
16	4.24	4.41	3.92	4.32	3.91	4.02	3.84	4.06
17	3.99	3.68	3.76	3.83	3.9	3.64	4.11	3.96
18	3.72	3.97	3.82	4.52	4.36	3.88	3.81	4.27
19	4.07	4.1	3.82	4.43	4.3	4.01	3.95	4.01
20	4.23	4.12	4.08	3.92	4.09	3.98	4	4.04
21	4.17	3.97	3.79	4.38	4.17	4.19	3.69	4.07
22	4.3	3.78	4.11	3.98	4.23	3.77	3.96	4.35
23	3.84	3.9	3.75	3.85	4.04	4.01	4.01	4.27
24	3.56	3.75	3.94	3.96	4.27	3.83	4.17	4.31
25	4.17	3.82	3.7	4.61	3.87	3.77	3.74	4.24
26	3.73	3.93	4.1	3.74	3.89	3.9	3.72	4.27
27	4.01	3.79	3.89	3.73	3.63	4.21	3.68	4.01
28	4.09	4.05	3.94	4.17	3.85	3.61	3.86	4.4
29	4.46	3.81	3.84	4.07	4.16	3.97	3.76	4.07
30	3.81	3.58	3.82	3.92	4.24	3.87	3.76	4
31	3.68	3.86	3.79	4.41	3.9	3.79	3.83	3.88
32	4.02	3.97	3.57	3.97	4.26	3.65	3.87	3.83
33	4.18	3.9	3.8	4.11	4.29	4.01	4.1	3.72
34	4.02	3.91	3.92	4.05	3.93	4.01	3.94	4.39
35	4.01	3.36	4.11	3.96	4.03	3.79	3.66	3.82
36	3.81	4.18	3.97	3.93	4.01	3.49	4.15	4.04
37	3.72	4.12	3.95	3.91	3.69	3.83	4.06	4.01
38	3.93	4.42	3.83	3.79	4.09	4.11	4.23	3.78
39	4.08	3.79	3.95	3.96	3.87	4.06	4.33	4.12
40	3.94	3.76	3.57	4.36	3.71	3.91	3.69	4.15
41	3.57	4.03	4.14	4.48	3.96	4.01	3.68	3.82
42	4.59	3.98	3.74	4.03	4.06	3.89	4.06	4.17
43	3.97	3.82	4.12	3.71	3.67	3.94	4.09	4.07
44	3.99	3.84	3.78	3.9	4.04	3.92	3.88	3.81
45	3.74	4	4.14	3.66	3.95	3.82	3.94	3.72
46	3.66	3.85	4.08	3.91	4.02	3.95	3.83	4.08
47	4.39	4.21	3.94	4.1	3.87	4.03	4.06	3.92
48	4.36	3.58	4.12	4.19	4.04	3.75	3.81	4.27
49	3.84	3.87	3.73	3.96	4.06	3.89	3.65	4.1
50	4.08	4.21	3.78	4.24	3.76	3.94	4.24	3.99
Min	3.54	3.36	3.30	3.50	3.63	3.49	3.51	3.58
Max	4.59	4.42	4.23	4.61	4.53	4.27	4.33	4.40
Ave	4.01	3.93	3.89	3.99	4.04	3.91	3.89	4.04
Stdev	0.24	0.21	0.18	0.25	0.22	0.18	0.19	0.19

Appendix G: UMP semester1-200708 multi neighbourhood GDA results based on random ordering (*cont...*)

No.	Remove specified Nh						
	Nh1	Nh4	Nh7	Nh1&Nh4	Nh1&Nh7	Nh4&Nh7	Nh1, Nh4&Nh7
1	4.41	3.99	4.11	4.67	4.3	4.48	4.64
2	3.68	4.11	3.9	4.47	4.5	4.59	4.6
3	3.6	4.42	4	3.88	4.02	3.89	4.51
4	3.88	4.25	4.16	4.17	4.08	4.04	4.03
5	3.7	4.26	4.39	3.92	4.04	4.22	4.46
6	3.92	3.95	3.95	4.13	4.56	4.31	4.83
7	4.16	3.96	3.6	3.75	4.3	4.56	4.43
8	4.1	3.72	4.29	4.43	3.95	3.93	4.71
9	3.74	4.12	3.99	4.26	4.2	4.31	4.42
10	4.03	3.91	4.38	4.17	4.39	3.86	4.04
11	4.15	3.76	4.02	4.61	3.92	3.92	4.51
12	4.34	4.56	3.8	4.46	3.64	4.25	4.69
13	4.29	3.99	3.76	4.8	4.47	4.32	4.3
14	4.4	4.61	3.83	4.47	3.91	4.15	4.44
15	4.43	3.78	3.79	4.46	4.31	3.98	4.46
16	4.14	4.34	4.21	3.84	4.26	4.12	4.7
17	3.9	4.11	4.12	4.67	3.83	4.36	4.31
18	4.06	3.97	4.23	4.21	4.18	4.18	4.36
19	4.05	4.37	4.37	4.3	4	4.52	4.4
20	4.07	4.31	4.29	4.41	3.99	3.96	5
21	4.26	4.34	3.97	4.52	4.1	4.32	4.72
22	4.24	3.96	4.11	3.99	4.19	4.29	4.38
23	4.14	4.24	3.83	3.98	4.37	3.64	4.39
24	4.01	3.87	4.1	3.97	4.41	4	4.71
25	4.1	3.82	4.24	3.85	4.37	3.99	4.46
26	3.86	4.36	4.38	3.94	4.13	3.88	4.52
27	3.79	4.01	3.96	4.38	4.24	4.45	3.96
28	3.86	4.19	3.94	4.48	4.28	3.87	4.7
29	4.14	4.58	4	4.23	3.77	4.25	4.52
30	4.54	4.14	3.83	4.41	4.34	4.4	4.39
31	4.16	3.79	4.05	4.27	4.46	4.33	4.25
32	4.34	3.96	4.45	4.5	4.37	4.11	4.83
33	4.06	3.6	4.26	4.08	3.94	4.26	4.24
34	3.73	4	3.83	4.47	4.25	4.47	4.16
35	4.17	4.21	3.95	4.18	4.15	4.55	4.83
36	4	4	4.24	4.14	4.26	3.97	4.53
37	4.06	4.33	3.8	4.2	4.32	4.17	4.16
38	4.05	4.16	3.81	4.26	4.1	4.64	4.5
39	3.85	3.87	4.22	4.53	4.05	4.47	4.48
40	4.05	4.41	4.15	4.48	4	4.25	4.27
41	3.98	3.99	4.19	4.38	4.02	4.36	4.16
42	3.71	4.04	3.8	4.68	3.9	4.14	4.14
43	4.18	4.3	4.36	4.27	4	4.2	4.39
44	4.23	4.21	3.97	4.2	4.19	4.09	4.06
45	4.21	4.15	3.67	4.7	4.18	3.98	4.4
46	3.9	3.78	3.78	4.21	4.16	4.63	4.98
47	4.14	4.42	4.28	4.47	4	4.26	4.4
48	4.04	3.85	4.26	4.18	4.43	4.09	5.18
49	4.27	4.14	3.95	4	4.09	4.49	4.56
50	4.61	4.4	3.64	4.23	4.18	4.11	4.18
Min	3.60	3.60	3.60	3.75	3.64	3.64	3.96
Max	4.61	4.61	4.45	4.80	4.56	4.64	5.18
Ave	4.07	4.11	4.04	4.29	4.16	4.21	4.47
Stdev	0.23	0.24	0.22	0.25	0.20	0.23	0.26

Appendix H: UMP semester1-200708 multi neighbourhood GDA results based on specified ordering

No.	All Nh	Remove specified Nh						
		Nh5	Nh2	Nh10	Nh5&Nh2	Nh5&Nh10	Nh2&Nh10	Nh5, Nh2&Nh10
1	3.86	4.1	3.9	3.86	4.21	3.9	3.95	4.05
2	3.58	4.24	3.78	4.15	4.13	3.95	3.93	4.29
3	3.77	3.8	3.87	3.73	3.7	3.55	3.87	4.3
4	3.94	3.79	3.89	3.89	3.86	3.84	4.05	4.18
5	3.75	3.83	3.86	3.79	4.1	3.83	4.12	4.13
6	4.23	4.01	3.69	3.57	3.83	3.92	3.76	4.16
7	4.16	3.87	4.14	3.73	4.06	3.84	3.82	4.13
8	3.95	3.81	3.85	3.78	4.03	3.9	4.2	4.03
9	3.83	4.08	4.12	3.71	3.98	4.04	3.96	3.77
10	3.82	3.82	3.77	3.47	4	3.55	3.83	3.97
11	3.84	3.77	3.67	4.2	3.89	3.96	3.96	4.27
12	3.54	3.76	3.59	3.72	3.68	3.55	3.75	4.37
13	3.91	3.89	3.9	3.73	4.27	3.82	4.07	4.22
14	4.52	3.86	4.02	4.04	4.14	3.83	3.95	4.36
15	3.77	3.67	3.99	3.49	4.3	3.67	3.95	4.13
16	3.6	3.85	4.09	3.72	3.93	3.52	4.15	3.89
17	3.97	3.99	3.78	4.14	4.25	3.78	3.8	4.4
18	3.71	4.09	3.71	3.69	4	4.16	3.94	4.49
19	4.11	4.5	3.78	3.67	3.81	3.73	4.1	3.92
20	4.04	3.89	3.82	3.82	4.03	3.81	3.99	3.71
21	4.1	3.72	4.03	4.1	3.94	4.03	3.74	4.04
22	3.75	3.74	3.95	3.93	4.14	3.64	4.14	3.8
23	3.92	3.9	3.98	3.99	4.28	3.5	3.8	4.47
24	3.82	3.51	3.99	4.01	4.29	3.72	3.88	4.09
25	3.46	3.94	3.75	3.95	3.7	3.97	3.84	3.68
26	3.54	4.35	3.79	4.09	3.66	3.96	3.67	3.69
27	3.63	3.96	3.68	4	3.98	4.28	4.08	4.16
28	3.93	3.74	3.91	3.9	3.97	3.72	4.08	3.86
29	3.9	3.84	4.1	4.12	4.07	3.97	3.91	3.91
30	3.72	3.82	3.85	3.6	3.87	3.83	3.87	4.08
31	4.01	3.99	4.19	4.02	3.96	3.69	3.75	3.92
32	3.92	3.44	3.61	3.66	3.97	3.57	3.62	4.43
33	3.99	3.88	4.03	3.91	3.89	3.92	3.89	4.22
34	3.96	3.88	3.89	3.75	4.13	3.72	3.83	4.07
35	4.24	3.58	3.9	3.91	4.11	3.93	4.02	4.08
36	4.13	3.94	3.84	4.2	3.74	3.92	3.81	4.33
37	3.67	3.66	3.79	3.55	4.19	3.78	3.78	3.99
38	4.36	3.75	3.87	3.89	4.09	3.68	3.83	3.54
39	4.09	3.86	3.78	3.91	3.96	3.85	3.84	4.09
40	3.75	4.1	3.67	3.96	4.15	4.07	4.14	3.83
41	3.57	3.97	3.88	3.73	4.09	3.78	3.83	4.06
42	4.16	3.92	4.18	3.79	3.97	4.03	3.5	4.47
43	3.9	3.87	3.83	4.02	4.66	3.91	3.94	3.81
44	4.23	3.78	3.87	4.21	3.9	4.11	3.96	4
45	3.65	3.8	4.04	4.11	4.32	4.11	3.71	4.07
46	4.03	3.55	3.63	3.93	3.92	3.91	3.82	3.67
47	4.59	3.9	3.98	4.16	3.73	3.99	3.73	4
48	3.77	3.76	3.77	4.35	3.94	3.89	4	3.67
49	3.7	4	3.41	3.97	3.89	3.97	3.82	4.17
50	3.82	4.12	3.75	4.01	4.03	4	3.91	4.07
Min	3.46	3.44	3.41	3.47	3.66	3.50	3.50	3.54
Max	4.59	4.50	4.19	4.35	4.66	4.28	4.20	4.49
Ave	3.90	3.88	3.86	3.89	4.01	3.85	3.90	4.06
Stdev	0.25	0.19	0.16	0.20	0.19	0.18	0.15	0.23

Appendix H: UMP semester1-200708 multi neighbourhood GDA results based on specified ordering (*cont...*)

No.	Remove specified Nh						
	Nh1	Nh4	Nh7	Nh1&Nh4	Nh1&Nh7	Nh4&Nh7	Nh1, Nh4&Nh7
1	4.13	3.96	4.01	4.23	4.28	3.96	4.12
2	4	4.05	4.02	4.22	4.27	4	4.24
3	3.91	3.48	3.95	4.21	4.12	4.21	4.17
4	3.89	3.92	4.01	5.04	4.21	3.86	4.22
5	4.07	4.03	3.66	4.23	3.65	3.98	4.59
6	3.95	3.8	3.84	4.26	4.26	4.1	4.43
7	3.89	3.75	4.03	4.41	4.1	3.98	4.3
8	3.85	3.79	3.81	4.1	3.93	3.93	3.8
9	3.94	3.91	4.06	3.94	4.36	4.03	4.19
10	4.12	3.91	3.91	4.25	4.51	4.21	4.28
11	3.91	4.13	3.93	4.07	4.09	4.01	4.25
12	3.9	4.32	3.64	4.21	4.2	4.05	4.15
13	3.89	3.96	3.85	4.1	4.51	3.87	4.48
14	4.03	4.27	4.18	4.02	3.96	4.13	3.75
15	3.82	4.06	3.95	4.07	4.2	4.18	4.04
16	4.21	3.94	4.54	4.3	3.7	3.79	4.14
17	4	3.91	4.2	4.07	3.87	3.99	4.55
18	3.69	3.59	3.62	4.06	3.7	4.29	4.21
19	4.17	3.81	4.56	4.55	3.98	4.14	4.42
20	4.01	4.18	3.73	4.32	4.28	3.8	4.12
21	4.34	3.92	3.99	4.44	4.41	4.01	3.92
22	3.97	3.93	4.1	4.46	4.06	3.54	4.33
23	3.93	3.88	3.9	4.03	3.9	4	4.19
24	3.88	4.23	3.92	3.74	3.8	3.92	4.52
25	4.34	3.62	4.04	4.3	3.93	3.93	4.39
26	3.95	4.12	3.93	3.98	3.94	3.75	4.35
27	4.46	4.14	4.05	4.14	4.08	3.97	4.07
28	3.98	4.37	4.02	4.23	4.16	3.97	4.32
29	4	3.93	4.06	4	4.16	3.88	4.28
30	4.03	3.9	3.66	4.28	4.29	3.9	4.22
31	3.67	3.88	4.2	4.39	3.95	3.83	4.43
32	4.32	4.01	4.04	4.06	4.01	3.97	4.12
33	3.89	3.85	4.11	4.23	3.8	4.12	4.08
34	4.16	3.96	3.74	3.95	3.69	4.16	4.87
35	3.93	4.11	3.93	4.4	4.26	4.21	4.39
36	3.77	3.86	3.97	4.2	4.22	4.05	4.61
37	3.96	3.66	4.1	3.81	4.24	4.29	4.16
38	3.8	4.09	3.89	4.11	3.82	3.91	4.39
39	3.72	3.69	3.7	4.68	4.19	3.98	4.32
40	3.96	4.12	4.1	3.99	4.1	3.78	4.33
41	4.25	4.07	3.84	4.18	3.95	4.46	4.3
42	3.99	4.4	3.56	3.92	4.03	3.95	4.14
43	4.06	4.1	4.01	4.5	3.82	3.84	4.49
44	4.19	3.91	3.55	4.45	4.21	4.38	4.73
45	3.99	4.28	4.11	3.88	4.16	3.91	4.61
46	3.99	3.99	4.17	4.16	3.98	3.95	4.72
47	4.2	4.36	4	3.97	3.98	4.14	4.46
48	3.79	4.22	3.9	4.36	3.68	3.81	4.48
49	4.21	3.85	3.87	4.18	4.27	3.78	4.41
50	3.86	3.48	4.3	4.58	4.07	3.8	4.44
Min	3.67	3.48	3.55	3.74	3.65	3.54	3.75
Max	4.46	4.40	4.56	5.04	4.51	4.46	4.87
Ave	4.00	3.97	3.97	4.21	4.07	3.99	4.31
Stdev	0.17	0.22	0.21	0.24	0.21	0.17	0.22

Appendix I: UMP semester1-200809 multi neighbourhood GDA results based on random ordering

No.	All Nh	Remove specified Nh						
		Nh5	Nh2	Nh6	Nh5&Nh2	Nh5&Nh6	Nh2&Nh6	Nh5, Nh2&Nh6
1	5.88	6.09	5.75	6.42	5.78	5.97	5.92	5.74
2	6.27	6.23	5.67	6.4	6	6	6.18	5.96
3	6.21	5.85	6.17	6.09	6.28	6.06	5.96	6.25
4	6.22	5.8	5.98	6.4	5.84	6.09	5.52	6.21
5	5.82	5.76	6.01	6.32	6.05	6.03	6.09	6.24
6	5.58	6.05	6.21	6.18	6.12	6.26	6.36	6.19
7	6.05	6.08	6.08	6.22	6.12	6.22	6.2	6.02
8	5.83	6.28	6.38	5.87	5.85	5.68	6.02	6.05
9	6	6.36	6.32	6.08	6.2	5.78	6	6.29
10	6.33	6.1	5.75	6.12	5.89	6.24	5.92	6.02
11	6.55	5.93	6.29	5.91	6.12	6.12	6	6.07
12	6.22	6.01	5.83	6.33	6.07	6.02	6.11	6.18
13	6.11	5.56	6.03	6.05	6.17	5.95	6.3	6.33
14	6.34	5.99	5.85	6.52	6.13	6.09	5.95	6.22
15	5.86	5.75	6.2	5.98	5.95	5.8	6.48	6.13
16	6.22	5.94	5.79	6.43	6.13	6.27	6.33	6.14
17	6.67	6.27	5.89	6.07	6.1	5.93	6.15	6.16
18	5.99	5.96	5.81	6.26	5.68	5.78	6.22	6.02
19	5.86	5.81	6.06	6.48	5.98	5.84	5.82	6.33
20	5.93	6.1	5.89	5.87	6.05	6.18	6.26	5.97
21	6.15	5.84	5.98	5.99	5.93	6.05	6.19	5.96
22	6.29	5.95	6.02	6.12	6	6.04	5.76	5.95
23	6.1	6.11	5.9	5.89	6.06	5.73	6.06	5.79
24	6.42	6.01	5.73	5.89	5.8	5.95	6.05	6.12
25	6.07	5.96	6.18	6.45	6.03	5.98	6.29	6.05
26	6.39	6.04	5.92	6.23	6.23	5.82	6.05	6.04
27	6.34	5.78	5.66	6.22	6.12	6.21	6.15	5.99
28	6.18	5.89	5.78	6.26	5.85	5.7	5.95	5.97
29	5.88	5.93	5.89	6.04	6.16	5.8	6.15	5.98
30	6.27	6.3	6.08	6.22	5.93	6.16	6.01	5.76
31	6.37	6.09	5.96	6.28	6.14	5.78	5.88	6.08
32	5.74	6.05	6.1	5.87	5.94	5.91	6.18	6.13
33	6.11	5.94	5.98	6.31	5.92	6.12	5.94	6.06
34	6.22	6.22	6.1	6.16	6.13	6.28	5.86	6.04
35	5.96	5.89	6.31	6.13	5.93	6.11	6.11	5.99
36	5.94	5.85	6.22	6.38	6.06	6.41	6.44	5.98
37	6.1	5.75	5.81	6.04	5.8	6.35	6.25	5.83
38	6.04	5.81	5.91	6.08	6.06	6	6.43	5.94
39	6.2	6.01	6.11	6.18	5.98	6.25	5.86	6.22
40	5.89	6.14	6.21	5.7	6.02	6.22	5.71	6.02
41	6.33	5.71	6.24	6.5	6.03	6.38	6.18	5.91
42	5.86	5.86	6.31	5.98	6.03	6.23	6.18	5.84
43	6.02	6.09	6	6.08	6.13	5.93	6.3	6.34
44	6.19	6.1	5.87	6.16	6.22	6.24	5.94	6.28
45	5.83	5.91	6.16	6.01	6.14	6.06	6.23	5.88
46	5.83	5.98	5.99	5.84	6.28	5.89	6.12	6.07
47	5.82	6.06	6.21	6.2	6.06	6.06	6.23	5.85
48	6.02	5.89	6.08	6.45	5.91	5.96	5.88	6.19
49	6.16	6.14	5.95	6.16	6.15	5.81	6.16	5.95
50	6.11	5.82	6.09	6.11	6.19	6.06	5.99	5.67
Min	5.58	5.56	5.66	5.70	5.68	5.68	5.52	5.67
Max	6.67	6.36	6.38	6.52	6.28	6.41	6.48	6.34
Ave	6.10	5.98	6.01	6.16	6.03	6.04	6.09	6.05
Stdev	0.22	0.17	0.18	0.20	0.14	0.19	0.20	0.16

Appendix I: UMP semester1-200809 multi neighbourhood GDA results based on random ordering (*cont...*)

No.	Remove specified Nh						
	Nh1	Nh3	Nh7	Nh1&Nh3	Nh1&Nh7	Nh3&Nh7	Nh1, Nh3&Nh7
1	6.21	6.36	6.21	6.39	6.61	6.36	6.91
2	6.57	6.38	6.44	6.65	6.41	6.47	6.91
3	6.42	6.3	6.26	6.92	6.7	6.3	7.18
4	6.36	6.18	6.53	6.41	6.52	6.51	7.16
5	6.33	6.52	6.18	6.31	6.49	6.67	7.29
6	6.18	6.36	6.16	6.37	6.08	6.64	7.03
7	6.12	6.23	6.04	6.59	6.39	6.64	6.98
8	6.51	6.48	6.12	6.49	6.62	6.71	7.01
9	6.29	6.27	6.01	6.93	6.41	6.65	6.99
10	6.54	6.33	6.21	6.65	6.72	6.44	7.06
11	6.28	6.02	6.35	6.4	6.27	6.18	6.88
12	6.13	6.44	6.07	6.62	6.43	6.54	6.84
13	6.25	5.96	6.13	6.54	6.3	6.61	7.09
14	6.16	6.62	6.29	6.39	6.35	6.32	6.91
15	6.12	6.04	6.13	6.54	6.58	6.53	7.08
16	6.54	6.52	6.37	6.27	6.75	6.3	7.12
17	6.15	6.21	5.87	6.69	6.33	6.48	7.02
18	6.26	6.06	6.01	6.33	6.64	6.54	6.97
19	5.77	6.62	6.36	6.24	6.52	6.55	7.17
20	6.16	6.72	6.35	6.77	6.75	6.56	7.11
21	6.29	6.39	6.46	6.54	6.42	6.32	7.03
22	6.47	6.59	6.13	6.31	6.63	6.2	6.71
23	6.32	6.47	6.16	6.35	6.62	6.48	7.29
24	6.37	5.97	6.37	6.58	6.6	6.54	6.99
25	6.48	6.44	5.8	6.68	6.68	6.63	6.72
26	6.33	6.49	6.41	6.72	6.43	6.31	7.04
27	6.57	6.19	6.42	6.58	6.62	6.19	7.27
28	6.15	6.53	6.17	6.59	6.44	6.22	7.14
29	6.42	6.12	6.03	6.26	6.45	6.09	6.69
30	6.18	6.01	6.23	6.4	6.37	6.51	7.03
31	6.14	6.67	5.95	6.44	6.55	6.78	6.76
32	6.02	6.35	6.35	6.58	6.11	6.32	6.81
33	6.41	6.13	6.65	6.58	6.47	6.48	7.37
34	5.76	6.3	6.37	6.5	6.95	6.63	7.05
35	6.56	6.47	6.05	6.65	6.32	6.2	7
36	6.5	6.26	6.13	6.92	6.17	6.53	6.88
37	6.2	6.28	6.07	6.49	6.8	6.36	7.49
38	6.22	6.52	6.55	6.55	6.47	6.06	7.29
39	6.35	5.99	6.18	6.73	6.58	6.37	7.04
40	6.43	6.3	6.37	6.94	6.56	6.2	6.9
41	5.89	6.36	6.03	6.44	6.52	6.55	6.85
42	6.43	5.9	6.46	6.69	6.74	6.38	7.32
43	5.98	6.3	6.25	6.53	6.35	6.5	7.11
44	6.29	5.82	6.38	6.54	6.22	6.17	6.68
45	6.08	6.03	6.15	6.78	6.4	6.39	7.12
46	6.39	6.21	6.11	6.26	6.67	6.64	7.22
47	6.67	6.21	6.1	6.4	6.47	6.43	7
48	5.98	6.34	6.06	6.75	6.5	5.92	7.42
49	6.22	6.65	6.2	6.94	6.69	6.18	6.91
50	6.38	5.6	6.27	6.55	6.64	6.24	7.11
Min	5.76	5.60	5.80	6.24	6.08	5.92	6.68
Max	6.67	6.72	6.65	6.94	6.95	6.78	7.49
Ave	6.28	6.29	6.22	6.56	6.51	6.42	7.04
Stdev	0.20	0.24	0.18	0.19	0.18	0.19	0.19

Appendix J: UMP semester1-200809 multi neighbourhood GDA results based on specified ordering

No.	All Nh	Remove specified Nh						
		Nh5	Nh2	Nh6	Nh5&Nh2	Nh5&Nh6	Nh2&Nh6	Nh5, Nh2&Nh6
1	6.21	5.88	6.3	6.18	5.89	6.13	5.92	6.18
2	5.98	5.92	5.92	5.88	6.22	6.39	6.04	6.03
3	6.18	5.88	6.05	6	6.06	6.49	6.2	6.11
4	5.97	6.1	5.74	5.99	5.75	5.87	5.9	5.9
5	6.15	5.84	6.08	6.02	6.25	5.81	6.04	6.25
6	6.24	6.04	6.07	5.78	5.6	6.1	5.58	6.27
7	6.34	6.03	6.21	5.8	5.79	6.04	5.95	6.08
8	6.02	6.13	5.82	6.03	6.06	6.22	6.12	6.05
9	6.05	5.79	6.21	6.23	5.93	5.78	6.11	6.19
10	5.94	5.73	5.73	6.04	6.11	6.01	5.97	6.17
11	6.04	5.95	6.08	6.15	6.08	6.43	6	5.76
12	6.01	6.22	5.88	6.1	5.96	5.79	6.33	5.89
13	5.94	6.01	5.96	6.12	6.3	6.19	5.99	6
14	6.09	5.83	5.98	5.75	6.24	6.07	6.08	6.43
15	6.12	5.55	6.12	5.73	5.75	5.89	6.06	6.25
16	6.03	5.84	5.93	6.19	6.2	6.44	6.27	5.91
17	6.24	6.04	5.77	6.14	6.19	6.13	6.08	6.33
18	6.15	5.68	5.95	6.07	6.14	6.03	6.11	6.08
19	5.99	5.78	5.78	6.2	5.99	6.11	5.79	5.87
20	6.04	6.01	5.77	5.89	6.07	5.62	6.24	6.06
21	6.03	5.95	5.77	5.8	6.15	6	6.24	6.19
22	5.71	6.17	5.99	6.25	5.97	6.25	6.08	6.34
23	5.96	5.81	5.89	5.95	6.21	5.98	5.88	6.2
24	6.1	6.14	5.97	6.11	5.78	6.12	6.05	5.94
25	6.4	6.16	5.87	6.22	5.74	6.14	6.01	6.18
26	5.87	6.09	6.32	6.06	5.96	6.08	5.78	6.02
27	5.92	5.95	6.2	5.94	6.34	6.05	5.81	6.14
28	5.89	5.95	6.05	5.86	5.95	6.37	5.96	5.86
29	6.01	6.16	6.03	5.8	6.11	6.09	6.03	5.98
30	6.09	6.22	6.03	5.92	6.27	5.9	6.14	6.04
31	6.14	5.92	5.77	6.38	6.21	5.87	5.93	6.26
32	6	6.05	5.88	6.26	6.16	5.75	6.05	6.03
33	6.28	6.04	6.08	6.02	6.12	6.04	6.18	6.09
34	6.06	6.05	5.94	5.91	6.02	5.97	5.92	6.06
35	6.14	6.18	5.82	6.1	6.14	5.85	6.17	5.82
36	6.18	5.81	6.05	6.01	6.28	5.85	5.92	6.12
37	5.93	5.88	6.14	5.95	5.85	5.86	6.02	6.21
38	6.02	5.75	6.47	6.1	6.16	5.62	6.2	5.84
39	6	5.81	5.53	6.13	6.18	6.06	5.98	6.2
40	5.74	5.99	6.27	6.04	6.08	6.19	5.76	6.24
41	5.92	6.02	6.14	6.09	5.74	5.94	5.95	6.02
42	6.18	6.09	6.28	5.8	6.11	5.69	6.15	6.24
43	5.92	5.84	5.84	6.01	6.24	6.16	6.24	5.75
44	5.88	6.12	5.93	6.12	6.53	5.58	5.92	5.9
45	6.05	6.1	5.9	5.85	6.22	6.19	5.59	5.98
46	6.35	6.16	6.12	5.91	5.98	6.21	6.2	6.41
47	6.23	6.19	5.81	5.81	5.85	5.65	5.62	6.05
48	6.25	5.63	5.78	5.81	6.03	6.02	6.02	6.01
49	6.09	5.88	6.22	5.96	6.16	6.23	6.3	6.11
50	6.14	6.18	5.81	5.83	6.23	6.03	6.02	5.97
Min	5.71	5.55	5.53	5.73	5.60	5.58	5.58	5.75
Max	6.40	6.22	6.47	6.38	6.53	6.49	6.33	6.43
Ave	6.06	5.97	5.99	6.01	6.07	6.03	6.02	6.08
Stdev	0.15	0.16	0.19	0.16	0.19	0.22	0.17	0.16

Appendix J: UMP semester1-200809 multi neighbourhood GDA results based on specified ordering (*cont...*)

No.	Remove specified Nh						
	Nh1	Nh3	Nh7	Nh1&Nh3	Nh1&Nh7	Nh3&Nh7	Nh1, Nh3&Nh7
1	6	5.8	6.34	6.57	6.22	6.17	6.98
2	6.24	5.93	6.13	6.62	6.5	6.09	7
3	6.3	6.08	5.91	6.55	6.45	6.14	7.27
4	6.47	6.46	6.11	6.77	6.8	6.41	6.86
5	6.39	6.26	5.9	6.55	6.19	6.2	7.14
6	6.56	6.22	6.01	6.22	6.6	6.36	7.07
7	6.2	6.51	6.45	6.45	6.35	6.2	7.21
8	6.02	6.35	5.93	6.38	6.55	6.16	7.29
9	6.2	6.27	6.27	6.52	6.32	6.71	7.3
10	5.85	6.01	6.4	6.66	6.68	6.14	7.39
11	6.17	6.3	5.87	6.71	6.5	6.27	7.1
12	6.05	6.46	6.2	6.49	6.45	6.41	7.12
13	6.28	6.33	5.84	6.5	6.3	6.62	6.87
14	6.07	6.23	6.2	6.48	6.14	6.42	6.76
15	6.14	6.14	6.14	6.51	6.42	6.48	7.18
16	6.22	6.17	5.95	6.75	6.65	6.3	6.73
17	6.29	5.88	6.05	6.25	6.32	6.31	6.92
18	6.02	6.03	6.19	6.91	6.3	6.21	7.28
19	6.49	5.78	6.12	6.47	6.54	6.48	7.11
20	6.56	6.23	6.32	6.36	6.34	6.26	7.34
21	6.44	6.45	5.98	6.59	6.34	6.17	7.2
22	6.23	5.92	5.98	6.45	6.85	6.44	6.9
23	6.07	6	6.18	6.05	6.39	6.51	6.78
24	6.17	6.39	6.24	6.5	6.52	6.41	6.91
25	6.21	6.24	6.08	6.55	6.16	5.94	7.14
26	5.95	6.1	6.28	6.56	6.46	6.28	6.63
27	6.14	6.42	5.87	6.61	6.85	6.49	6.99
28	6.05	6.02	6.24	6.48	6.52	6.4	6.93
29	6.34	5.93	6.13	6.55	6.48	6.3	7.07
30	6.13	6.09	6.08	6.3	6.64	6.35	6.9
31	5.99	6.42	6.01	6.25	6.49	6.46	7.23
32	5.95	6.05	6.2	6.41	6.37	6.02	7.59
33	6.39	6.2	6	6.81	6.19	6.44	7.19
34	6.27	6.21	6.16	6.44	6.44	6.09	7.46
35	6.15	6.36	6.2	6.44	6.69	5.99	7.17
36	5.88	6.07	6.34	6.39	6.4	6.22	6.81
37	5.93	6.31	6.05	6.65	6.73	6.31	6.9
38	6.24	5.98	6.12	6.7	6.69	6.19	7.01
39	6.26	6.04	6.24	5.95	6.08	6.33	6.67
40	6.15	6.12	6.51	6.51	6.58	6.01	7.16
41	6.63	5.85	6.32	6.4	6.65	6.18	7.12
42	5.93	6.44	5.98	6.42	6.44	6.25	7.08
43	6.17	6.02	6.27	6.41	6.5	6.33	7.11
44	6.36	6.02	6.32	6.36	6.71	6.36	7.21
45	6.64	6.07	5.93	6.6	6.2	6.59	6.98
46	6.54	5.95	6.05	6.46	6.57	6.27	7.11
47	5.92	6.24	5.91	6.41	6.61	6.12	7.27
48	6.69	6.08	5.85	6.6	6.49	5.98	7.09
49	5.75	6.39	6.01	6.08	6.24	6.34	6.99
50	6.2	6.19	6.13	6.63	6.3	6.4	7
Min	5.75	5.78	5.84	5.95	6.08	5.94	6.63
Max	6.69	6.51	6.51	6.91	6.85	6.71	7.59
Ave	6.21	6.16	6.12	6.49	6.46	6.29	7.07
Stdev	0.22	0.19	0.17	0.18	0.19	0.17	0.20

Appendix K

Table 7.7a. *p*-value result for semester1-200708 in comparison between the neighbourhood heuristic (random ordeing)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a) All Nh	-	.871	.211	1.00	1.00	.481	.279	1.00	.990	.743	1.00	.000	.064	.005	.000
b) Remove Nh5	.871	-	.999	.986	.372	1.00	1.00	.320	.074	.008	.347	.000	.000	.000	.000
c) Remove Nh2	.211	.999	-	.550	.021	1.00	1.00	.011	.002	.000	.020	.000	.000	.000	.000
d) Remove Nh10	1.00	.986	.550	-	1.00	.824	.620	1.00	.936	.534	.999	.000	.030	.002	.000
e) Remove Nh5 and Nh2	1.00	.372	.021	1.00	-	.079	.033	1.00	1.00	.964	1.00	.000	.216	.019	.000
f) Remove Nh5 and Nh10	.481	1.00	1.00	.824	.079	-	1.00	.050	.008	.001	.074	.000	.000	.000	.000
g) Remove Nh2 and Nh10	.279	1.00	1.00	.620	.033	1.00	-	.019	.003	.000	.031	.000	.000	.000	.000
h) Remove Nh5, Nh2 and Nh10	1.00	.320	.011	1.00	1.00	.050	.019	-	1.00	.911	1.00	.000	.100	.006	.000
i) Remove Nh1	.990	.074	.002	.936	1.00	.008	.003	1.00	-	1.00	1.00	.002	.766	.176	.000
j) Remove Nh4	.743	.008	.000	.534	.964	.001	.000	.911	1.00	-	.978	.047	.998	.730	.000
k) Remove Nh7	1.00	.347	.020	.999	1.00	.074	.031	1.00	1.00	.978	-	.000	.279	.028	.000
l) Remove Nh1 and Nh4	.000	.000	.000	.000	0.00	.000	.000	0.00	.002	.047	.000	-	.322	.974	.046
m) Remove Nh1 and Nh7	.064	.000	.000	.030	.216	.000	.000	.100	.766	.998	.279	.322	-	.998	.000
n) Remove Nh4 and Nh7	.005	.000	.000	.002	.019	.000	.000	.006	.176	.730	.028	.974	.998	-	.000
o) Remove Nh1, Nh4 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.046	.000	.000	-

Table 7.7b. *p*-value result for semester1-200708 in comparison between the neighbourhood heuristic (specified ordering)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a) All Nh	-	1.00	1.00	1.00	.458	.996	1.00	.093	.642	.973	.991	.000	.045	.730	.000
b) Remove Nh5	1.00	-	1.00	1.00	.044	1.00	1.00	.004	.082	.566	.693	.000	.001	.121	.000
c) Remove Nh2	1.00	1.00	-	1.00	.004	1.00	.998	.000	.008	.219	.313	.000	.000	.015	.000
d) Remove Nh10	1.00	1.00	1.00	-	.149	.999	1.00	.018	.257	.831	.912	.000	.006	.341	.000
e) Remove Nh5 and Nh2	.458	.044	.004	.149	-	.003	.065	.999	1.00	1.00	.996	.002	.994	1.00	.000
f) Remove Nh5 and Nh10	.996	1.00	1.00	.999	.003	-	.984	.000	.005	.140	.207	.000	.000	.008	.000
g) Remove Nh2 and Nh10	1.00	1.00	.998	1.00	.065	.984	-	.006	.119	.759	.868	.000	.001	.181	.000
h) Remove Nh5, Nh2 and Nh10	.093	.004	.000	.018	.999	.000	.006	-	.975	.837	.702	.147	1.00	.951	.000
i) Remove Nh1	.642	.082	.008	.257	1.00	.005	.119	.975	-	1.00	1.00	.000	.919	1.00	.000
j) Remove Nh4	.973	.566	.219	.831	1.00	.140	.759	.837	1.00	-	1.00	.000	.697	1.00	.000
k) Remove Nh7	.991	.693	.313	.912	.996	.207	.868	.702	1.00	1.00	-	.000	.528	1.00	.000
l) Remove Nh1 and Nh4	.000	.000	.000	.000	.002	.000	.000	.147	.000	.000	.000	-	.146	.000	.597
m) Remove Nh1 and Nh7	.045	.001	.000	.006	.994	.000	.001	1.00	.919	.697	.528	.146	-	.865	.000
n) Remove Nh4 and Nh7	.730	.121	.015	.341	1.00	.008	.181	.951	1.00	1.00	1.00	.000	.865	-	.000
o) Remove Nh1, Nh4 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.597	.000	.000	-

Table 7.8a. *p*-value result for semester1-2008/09 in comparison between the neighbourhood heuristic (random ordering)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a) All Nh	-	.211	.798	.973	.942	.980	1.00	.996	.004	.005	.149	.000	.000	.000	.000
b) Remove Nh5	.211	-	1.00	.000	.905	.965	.219	.767	.000	.000	.000	.000	.000	.000	.000
c) Remove Nh2	.798	1.00	-	.021	1.00	1.00	.848	1.00	.000	.000	.000	.000	.000	.000	.000
d) Remove Nh6	.973	.000	.021	-	.032	.112	.878	.148	.197	.173	.955	.000	.000	.000	.000
e) Remove Nh5 and Nh2	.942	.905	1.00	.032	-	1.00	.968	1.00	.000	.000	.000	.000	.000	.000	.000
f) Remove Nh5 and Nh6	.980	.965	1.00	.112	1.00	-	.992	1.00	.000	.000	.000	.000	.000	.000	.000
g) Remove Nh2 and Nh6	1.00	.219	.848	.878	.968	.992	-	.999	.001	.001	.043	.000	.000	.000	.000
h) Remove Nh5, Nh2 and Nh6	.996	.767	1.00	.148	1.00	1.00	.999	-	.000	.000	.000	.000	.000	.000	.000
i) Remove Nh1	.004	.000	.000	.197	.000	.000	.001	.000	-	1.00	.973	.000	.000	.041	.000
j) Remove Nh3	.005	.000	.000	.173	.000	.000	.001	.000	1.00	-	.932	.000	.000	.206	.000
k) Remove Nh7	.149	.000	.000	.955	.000	.000	.043	.000	.973	.932	-	.000	.000	.000	.000
l) Remove Nh1 and Nh3	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	-	.991	.030	.000
m) Remove Nh1 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.991	-	.506	.000
n) Remove Nh3 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.041	.206	.000	.030	.506	-	.000
o) Remove Nh1, Nh3 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	-

Table 7.8b. *p*-value result for semester1-2008/09 in comparison between the neighbourhood heuristic (specified ordering)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a) All Nh	-	.170	.561	.827	1.00	.999	.981	1.00	.021	.251	.900	.000	.000	.000	.000
b) Remove Nh5	.170	-	1.00	.999	.308	.984	.986	.071	.000	.000	.002	.000	.000	.000	.000
c) Remove Nh2	.561	1.00	-	1.00	.683	1.00	1.00	.319	.000	.001	.020	.000	.000	.000	.000
d) Remove Nh6	.827	.999	1.00	-	.902	1.00	1.00	.559	.000	.002	.041	.000	.000	.000	.000
e) Remove Nh5 and Nh2	1.00	.308	.683	.902	-	.999	.989	1.00	.066	.479	.975	.000	.000	.000	.000
f) Remove Nh5 and Nh6	.999	.984	1.00	1.00	.999	-	1.00	.983	.007	.087	.501	.000	.000	.000	.000
g) Remove Nh2 and Nh6	.981	.986	1.00	1.00	.989	1.00	-	.867	.001	.014	.170	.000	.000	.000	.000
h) Remove Nh5, Nh2 and Nh6	1.00	.071	.319	.559	1.00	.983	.867	-	.096	.621	.996	.000	.000	.000	.000
i) Remove Nh1	.021	.000	.000	.000	.066	.007	.001	.096	-	.999	.662	.000	.000	.705	.000
j) Remove Nh3	.251	.000	.001	.002	.479	.087	.014	.621	.999	-	.998	.000	.000	.036	.000
k) Remove Nh7	.900	.002	.020	.041	.975	.501	.170	.996	.662	.998	-	.000	.000	.000	.000
l) Remove Nh1 and Nh3	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	-	1.00	.000	.000
m) Remove Nh1 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	1.00	-	.000	.000
n) Remove Nh3 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.705	.036	.000	.000	.000	-	.000
o) Remove Nh1, Nh3 and Nh7	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	-