A STUDY OF EXAMINATION TIMETABLE PROBLEM IN UMP

MOHD NIZAM MOHMAD KAHAR (PI)
KU SAIMAH BINTI KU IBRAHIM
SURYANTI BINTI AWANG
ZALILI BINTI MUSA
ROHANI BINTI ABU BAKAR
TUTY ASMAWATY BINTI ABDUL KADIR

RESEARCH VOTE NO:
RDU140383

Fakulti Sistem Komputer & Kejuruteraan Perisian
Universiti Malaysia Pahang

ABSTRACT

The examination timetabling problem involves the task of assigning the examinations into a limited number of timeslots and rooms with the aim of satisfying all the hard constraints. Most of the reported research in the literature starts with constructing the initial timetable by scheduling all the examinations and then performs an improvement on the timetable. In this research, we investigate a real world examination timetabling problem from Universiti Malaysia Pahang (UMP). UMP examination timetabling dataset is a capacitated dataset which contains additional constraints, in addition to those commonly used in the literature. The proposed algorithms start with constructing the initial timetable using the graph heuristic methods. The entire process runs until all of the examinations are assigned successfully. An improvement on the solution was implemented using step-count hill climbing and late acceptance hill climbing. The proposed approaches are tested on two benchmark datasets, namely Toronto dataset and the Universiti Malaysia Pahang (UMP) dataset. The experimental results show that the proposed approaches are able to produce good quality solution when compared to the solutions from the proprietary software used by UMP. Additionally, our solutions satisfy to all the hard constraints which the current systems fails to do.

ABSTRAK

Masalah perjadualan waktu peperiksaan melibatkan tugas menjadualkan peperiksaan ke dalam bilangan slot masa dan bilik yang terhad dengan tujuan untuk memenuhi semua kekangan keras. Kebanyakan kajian yang dilaporkan dalam kesusasteraan bermula dengan membina jadual waktu awalan kemudian melakukan peningkatan pada jadual waktu. Dalam kajian ini, kami mengkaji masalah pemeriksaan jadual waktu dunia sebenar dari Universiti Malaysia Pahang (UMP). UMP peperiksaan jadual waktu set data adalah set data berkapasiti yang mengandungi kekangan tambahan, tambahan kepada yang biasa digunakan dalam kesusasteraan. Algoritma yang dicadangkan bermula dengan membina jadual waktu awal menggunakan graf kaedah heuristik. Keseluruhan proses berjalan sehingga semua peperiksaan yang dijadualkan dengan jayanya. Peningkatan yang digunakan step counting hill climbing dan late acceptance hill climbing. Kaedah yang dicadangkan diuji pada dua set data penanda aras, iaitu Toronto set data dan set data Universiti Malaysia Pahang (UMP). Keputusan eksperimen menunjukkan bahawa kaedah yang dicadangkan dapat menghasilkan penyelesaian yang berkualiti baik jika dibandingkan dengan penyelesaian daripada perisian proprietari yang digunakan oleh UMP. Selain itu, penyelesaian kami memenuhi semua kekangan keras yang sistem semasa tidak berbuat.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter presents an introduction of the research work. Section 1.2 describes the background of the research and basic overview of the timetabling problem. Section 1.3 describes the problem statements and the research questions are listed in Section 1.4. Section 1.5 and 1.6 describe the objectives and scopes of the research respectively. Lastly, the report organization is describe in Section 1.7.

1.2 Background of Research

Research on timetabling on course and examination has been conducted over the years. Many academic institutions face a considerable amount of challenges in producing optimal course and examination timetable in a reasonable time within a limited resource. Both timetabling problems are similar in nature which involves assigning the courses or exams into available timeslots aim to satisfy varying types of constraints (Wren, 1996; Burke, Kingston and de Werra, 2004). Although the course and examination timetabling have many similarities, these two problems actually differ in terms of their constraints, user preferences and in the way the problem is constructed (McCollum et al., 2007; Abdullah & Turabieh, 2012; Pillay, 2016). For example, an examination timetable may allow multiple exams to be carried out in the same room while it is not possible to have two different courses in the same room. Examination timetabling is more difficult compared to course timetabling because the examinations are organized with registered students. Therefore, some constraints such as a clash free (hard constraint) examination timetabling and student satisfaction (soft constraint) have to be considered in producing good quality examination timetable.

This research concentrates on the examination timetabling problem. Like most timetabling problems, generating a good quality examination timetable is a challenging and time consuming task due to its combinatorial and highly constrained nature. Examination timetabling is a process of assigning examinations to a specific number of examination periods to satisfy the constraints. There are two categories of constraints that contribute to the complexity of an examination timetabling, namely hard

constraints and soft constraints. The hard and soft constraints need to be considered in producing the examination timetable because it reflects the institution need and requirement of their timetable. Hard constraints refer to those requirements which cannot be broken and must be satisfied at all the times. A timetable is considered feasible if it satisfy the hard constraints. An example of hard constraint is a student should not be required to sit two examinations at the same time, where the examination timetable should be clash free. Soft constraints refer to those requirements that are not essential but should be satisfied as much as possible. According to Ayob, Abdullah and Malik (2007), it is not possible to fulfil all the soft constraints, hence it is normally being used to evaluate the quality of the examination timetable by associating a weighted penalty value with each violation of the soft constraints. An example of a soft constraint is that exams need to be spread evenly throughout the examination period so that students have time for revision between exams.

The examination timetabling problem can be divided into two categories of problems, which is the un-capacitated problems and capacitated problems. The room capacities are not considered in the un-capacitated examination timetabling problem while in the capacitated examination timetabling problem the capacities of the room are considered as a hard constraint. The main challenge in examination timetabling is to allocate a large number of examinations within the limited resources, such as time periods and room availabilities (Burke, Newall and Weare, 1996). According to Burke et al., (1996), 73% of universities reported that accommodating examinations is a major problem. Capacitated problem resemble the real world problem and it is more complicated compared to un-capacitated problem. The room constraint increases the level of complexity to the overall problem in producing a good quality solution. Moreover, the increases of the number of student enrolments and constraints cause the examination timetabling becomes more challenging. The difficulty of the problem is added when these constraints conflict with one another where satisfaction of one constraint can lead to a violation of other constraints (Qu et al. 2009; Kahar and Kendall, 2010). For example, in such a situation in which we want to reduce the total number of timeslots and at the same time to spread the examinations as much as possible over the examination period.

In this research, we are concern with a real world examination timetabling problem from Universiti Malaysia Pahang (UMP) which has a number of different constraints from the literature. The additional hard constraints are the splitting of an examination into different rooms in the same building and students cannot be scheduled to an examination at different campus as UMP has two campus. Besides, the additional soft constraints are the minimization of the room distance for an examination in multiple rooms and the minimization of the number of rooms can be split across for an examination.

1.3 Problem Statements

The examination timetabling problems have attracted the interested of many researchers over the years. A lot of methods have been proposed in an attempt to produce a good quality solution. Constructing examination timetabling consists of two phases where the first phase is to generate a complete initial solution. The process continues by improving the initial solution timetable which is the second phase (Asmuni et al., 2009; Gogos et al., 2012; Kahar and Kendall, 2013). Normally, graph heuristics are being used in initial phase to generate the initial solution with the aim of satisfying the hard constraints. In the improvement stage, meta-heuristics are used to improve of the initial solution. Examples of this technique include hill climbing, tabu search, simulated annealing, genetic algorithm and other methods (detail discussion in chapter 2).

This research is concerned with a real world examination timetabling problem from Universiti Malaysia Pahang (UMP). UMP examination timetabling dataset is a capacitated dataset that contains additional constraints apart from the commonly used soft constraints. The additional hard constraints are,

a) The examinations need to be scheduled to the appropriate campus; students in campus Gambang are not allow to take the examination in campus Pekan and vice versa.
b) The examinations with the same code but from different campus need to be scheduled simultaneously; the examinations for university courses (compulsory for all students) should be scheduled at the same time. These constraints have not been investigated before in the literature.

The soft constraints are

a) The distance of the assigned rooms for an examination should be minimised
b) The number of rooms for a split examination should be minimised.

A comparison of the constraints is discussed in chapter 3. This motivates us to propose UMP datasets as benchmark problem instances and to propose an improved examination timetabling approach for UMP.

1.4 Research Questions

The research questions are stated for the research guidelines:

- Can we produce a feasible solution for UMP capacitated examination timetabling problem considering two campus constraints.

- Can we produce an optimal solution considering the new constraints in UMP examination timetabling problem?

1.5 Objectives

The objectives of the research are as follows:

- To investigate Universiti Malaysia Pahang (UMP) capacitated examination timetabling problem that has dual campus far apart and individual constraint from different UMP campus.
- To implement meta-heuristics approach in solving the UMP examination timetabling problem.
- To evaluate and compare the result of proposed approach with the examination timetable being used by UMP.

1.6 Scopes

The scopes of the work are as follows:

- The investigation concentrates on a real-world benchmark dataset which is University Malaysia Pahang (UMP) examination dataset (e.g. semester 1 and 2, 2014/2015).
- Investigation focuses on graph heuristics ordering strategy which include largest enrolment (LE), largest degree (LD), largest weighted degree (LWD) and saturation degree (SD). The improvement methods include hill climbing (HC).
- The research uses Delphi Program and Pascal language to develop the program for examination timetabling.
- The improvement methods include hill climbing and great deluge algorithm.

1.7 Report Organization

This thesis is organized into five chapters. Chapter 1 explains about the basic overview of the timetabling and examination timetabling, problem statements, research questions, research objective and the scope of the study. Chapter 2 describes great deluge algorithm in solving the UMP examination timetabling problem. Chapter 3 describes the formal mathematical model considering the newer constraints. Chapter 4 discusses the conclusion and the contribution of the research work.

CHAPTER 2

A GREAT DELUGE ALGORITHM FOR A REAL WORLD EXAMINATION TIMETABLING PROBLEM

Abstract

The examination timetabling problem involves assigning exams to a specific or limited number of timeslots and rooms, with the aim of satisfying all hard constraints (without compromise) and satisfying the soft constraints as far as possible. Most of the techniques reported in the literature have been applied to simplified examination benchmark datasets. In this paper we bridge the gap between research and practice by investigating a problem taken from the real world. This paper 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.

Keywords: Examination Timetabling, Great Deluge Algorithm, Scheduling

---

2.1 Introduction

Examination timetabling has been the subject of research studies for many years. The exam timetable generated by any institution has a significant impact on students, lecturers and administrators. The timetable should aim to satisfy all interested parties and hence the solution needs to take into account many factors such as ensuring that there are no clashes (i.e. two exams at the same time) for students, there is sufficient marking time for lecturers and that administrations are also happy with the timetable (McCollum, 2007). Many papers discussing the examination timetable problem can be found in the literature, with a good source being the PATAT conference series of selected papers (i.e. Burke

and Ross, 1996a; Burke and Carter, 1998; Burke and Erben, 2001; Burke and De Causmaecker, 2003; Burke and Trick, 2005; Burke and Rudova, 2007).

In this paper 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. The rest of the paper is organised as follows. In sections 2 and 3, we describe the examination timetabling problem and related work. In sections 4 and 5, we describe the GDA and our proposed modification. A description of the UMP examination timetabling problem, including the constraints, is discussed in section 6. In section 7, we describe the experimental setup to allow reproducibility for other researchers. The result from the improvement phase is shown in section 8, followed by statistical analysis in section 9. Discussion on the results is presented in section 10. Lastly, in sections 11 and 12 we summarise the contribution and present our conclusions.

2.2 Examination timetabling problem

The examination timetabling problem involves assigning exams to a limited number of timeslots and rooms with the aim of satisfying the *hard constraints* and minimising the violations of *soft constraints* (e.g., Carter and Laporte, 1996a; Qu et al., 2009). Hard constraints cannot be broken and a timetable is considered feasible if all the hard constraints are satisfied. An example of a hard constraint is that no student is assigned to sit more than one exam at the same time. Soft constraints are requirements that are not essential but should be satisfied as far as possible, hence, it is used to evaluate the quality of the solutions through (perhaps weighted) summation of the number of violations of the soft constraints. An example of a soft constraint is to spread exams as evenly as possible from a student's point of view. A list of commonly used constraints is given in Qu et al. (2009), Merlot et al. (2003), Burke et al. (1996c) and Cowling et al. (2002).

Examination timetabling problems can be categorised into un-capacitated or capacitated variants. Unlike the un-capacitated examination timetabling problem, in capacitated problems, room capacities are treated as a hard constraint (which more closely reflects real world problems), in addition to the other commonly used hard constraints (Pillay and Banzhaf, 2009; Abdullah, 2006). Most of the research seen in the literature has investigated the un-capacitated examination timetabling problem, the Toronto dataset (Carter et al., 1996b) being a good example. Research on this dataset has mainly focussed on the algorithmic performance to produce solutions effectively and quickly (see Qu et al., 2009). However, McCollum (2007), Qu et al. (2009) and Carter and Laporte (1996a) 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 problem, which only addresses a few common hard constraints (i.e. no exams with common students assigned simultaneously) and soft constraints (i.e. spreading conflicting exams as evenly as possible).

Capacitated problems more closely resemble real world problems as they include a room capacity constraint and Burke et al.'s (1996c) survey showed that 73% of universities agree that accommodating exams is a difficult problem. The difficulties are caused by (a) the lack of available exam rooms due to (for example) the room being used for teaching purposes and (b) the splitting of exams between more than one room. The size of an exam alone could easily cause the exams to be split across more than one room and this should result in additional constraints such as splitting exams onto different sites and/or the distance between rooms (Burke et al., 1996c).

However, the capacitated problem has received less attention from the research community probably due to the lack of benchmark datasets. Furthermore, it requires more comprehensive data as it has to include the room capacities and this information can be difficult to collect (McCollum, 2007). Examples of capacitated examination datasets available in the literature are the Nottingham dataset (Burke et al., 1996b), the Melbourne dataset (Merlot et al., 2003), the International Timetabling Competition (ITC2007) dataset (McCollum et al., 2010) and the Toronto dataset (which includes the total seating capacity introduced by Burke et al., 1996b).

The Nottingham, Melbourne and (*modified*-)Toronto dataset is concerned with the total seating capacity (allowing multiple exams in the same room). That is, the total number of students sitting in all exams, in the same timeslot, must be less than some specified number. This represents a simplified problem whereas normally in solving a real-world problem, we would have to take into account individual room capacities (Merlot et al., 2003), but this obviously depends on institutional requirements. The ITC2007 dataset provides more realistic problems than the (*modified*-)Toronto, Nottingham and Melbourne dataset problems, as it considers individual room capacities. However for the UMP dataset, besides considering individual room capacities, it does not allow multiple exams to

share a single room. This complicates the problem even further. A description of UMP constraints is described in section 6 (also see Kahar and Kendall, 2010a).

2.3 Related work

A variety of techniques have been utilised in solving the examination timetabling problem (see Carter and Laporte, 1996a; Schaerf, 1999 and Qu et al., 2009). One popular technique is Hill climbing (HC). HC accepts a candidate solution, if it is an improving solution. HC is simple and easy to implement, however it is easily trapped in local optima (Burke and Kendall, 2005). Recently, Burke and Bykov (2008) have proposed a late acceptance strategy for hill climbing. The method delays the comparison step between a candidate solution and the current (best) solution. The late acceptance hill climbing methodology is able to produce good quality solutions compared to other works for the Toronto datasets. Tabu search (Glover, 1986) works in a similar way to hill climbing but incorporates a memory. To prevent the search from becoming stuck in a local optima, a tabu list (memory) is used to hold recently seen solutions or, more likely, some of the attributes of the neighbourhood move and ignore solutions which are in the tabu list (Di Gaspero and Schaerf, 2001; Di Gaspero and Schaerf, 2002, White, Xie and Zonjic, 2004).

Simulated annealing (SA) (Kirkpatrick, 1983) always accepts better solutions but also accepts worse solutions with a decreasing probability as the search progresses (Merlot et al., 2003; Burke et al., 2003). Great Deluge (Dueck, 1993) works in almost the same way as SA but it uses a boundary as the acceptance criteria, which is gradually reduced. New solutions are only accepted if they improve on the current solution or are within the boundary value. A further discussion on this methodology follows in the next section.

Variable Neighbourhood Search (VNS) (Mladenovic and Hansen, 1999) is based on the strategy of using more than one neighbourhood structure and changing them systematically during the search. The use of many neighbourhoods allows VNS to more effectively explore the search space (Abdullah et al., 2005; Burke et al., 2010a etc). Other techniques in the literature include ant colony optimisation (ACO) (Dorigo, 1996). ACO is inspired by the behaviour of ants, and the way they forage for food. Only a few works using ACO for exam timetabling have been reported in the literature (Eley, 2006 and Eley, 2007).

Genetic Algorithms (GA) (Burke and Kendall, 2005) are a population based search which uses the principle of biological evolution (i.e. selection, mutation, crossover) to generate better solutions from one generation to another (Ross and Corne, 1995 and Burke et al., 2010a). Memetic algorithms (MA) are a hybridisation of GA, by incorporating a local-search algorithm. MAs have been shown to

work effectively compared to GAs (Burke, Newall and Weare, 1996b; Abdullah, Turabieh and McCollum, 2009 etc).

Hyper-Heuristics (HH) (Burke et al., 2010b) are methods that attempt to raise the level of generality at which search methodologies operate. HH can be categorised as a method that selects heuristics or generates new heuristics from components of existing heuristics (Burke et al., 2010b). HH are concerned with the exploration of a heuristic space rather than dealing directly with solutions to the problem (Kendall and Hussin, 2004; Hussin, 2005; Burke et al., 2007 etc).

Many other techniques can be found in the PATAT series of conference volumes.

2.4 The Great Deluge Algorithm

In 1993, Dueck introduced the great deluge algorithm (GDA) that operates in a similar way to simulated annealing. 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). GDA only involves one parameter (decay rate) which is an advantage over SA since the effectiveness of a meta-heuristic technique is often dependent upon parameter tuning (Petrovic and Burke, 2004).

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 0.01. Dueck was able to find good quality solutions. Burke and Newall (2002) implemented GDA in order to solve 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 reported that GDA was superior to both these 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 evaluation function that is required). 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). According to Burke et al., the time-predefined GDA was superior when compared to other methods. McMullan (2007), proposed the use of a steeper decay rate (with decay rate propotional to 50% of the entire run on the first stage and 25% on the remaining runs) compared to Burke et al. (2004), in order to force the algorithm to reach better quality

solutions as early as possible. He also allowed the algorithm to 'reheat' (similar to simulated annealing) allowing worse moves to be accepted. The re-heat mechanism is activated when there is no improvement. The method is able to find better solutions when compared to other methods (except on small instances).

Silva and Obit (2007) proposed a non-linear decay rate for the boundary control using an exponential function. They also included a feature that allows the boundary to rise when its value is about to converge to the penalty cost of the solution. Experimentation on the course timetabling problem revealed that the non-linear GDA outperforms some of the previous results. Abdullah et al. (2009) investigated the hybridisation of a great deluge algorithm and tabu search in solving the course timetabling problem. Their algorithm applied four neighbourhood moves (at every iteration) and selected the best move. If there is no improvement within a specified time, the boundary is increased by a value between zero and three, selected at random. The combination of the two meta-heuristics produced good results.

In this work, we utilise GDA for the UMP examination problem, a real world timetabling problem that contains several new constraints. Full details of the problem is given in section 6.

2.5 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 understand 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 officers as they are the persons responsible for producing the timetable at UMP. Our proposed GDA is shown in figure 1. 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). This is to allow acceptance of poorer solutions. We have tried several other percentages; a higher percentage leads to the search being unfocused, whilst a smaller percentage discourages exploration. Based on our observation, a value of 3% 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 by having a new desired value. The value of 0.8 is chosen to avoid having a steep boundary thus encouraging exploration of the search space. 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) and solution produced by UMP and from our previous work (Kahar and Kendall, 2010a).

2.6 University Malaysia Pahang (UMP): Examination Timetabling Problem

The Universiti Malaysia Pahang (UMP) was established in 2002 and currently operates from a temporary campus. This presents many challenges in terms of available space, logistics and human resources to manage the university operation. Furthermore, new faculties are being introduced along with new programs being offered. This results in an increase in the number of students which, according to the timetable officer, makes it difficult to generate a feasible examination timetable. In addition to these limitations, the UMP examination timetable problem has other challenging constraints. The hard and the soft constraints for the UMP *examination assignment* are listed in table 1. Further details and the mathematical model are available in Kahar and Kendall (2010a).

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
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 2.1 Our proposed Great Deluge Algorithm

Table 2.1 The exam timetabling constraints

Hard constraints	
1.	No student should be required to sit two examinations simultaneously.
2.	The total number of students assigned to a particular room(s) must be less that the total room capacity.
3.	Exams which have been split into several rooms must be in the same building.
4.	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).
Soft Constraints	
5.	Each set of student examinations should be spread as evenly as possible over the exam period.
6.	The distance between exam rooms, for the same exam, should be as close as possible to each other (and within the same building).
7.	A penalty value is applied when splitting an exam across several rooms, as we prefer an exam to be in a single (or as few as possible) room whenever possible.

2.7 UMP Examination Timetabling dataset

Our investigations were carried out using two different datasets from semester1-2007/08 and semester1-2008/09. In semester 1-2007/08 the total number of examination papers is 157, across 17 programs offered by 5 faculties. The total number of students is 3,550 with 12,731 enrolments. The conflict density matrix is 0.05, which means that 5% of students are in conflict among the examination papers. The total available exam space for this dataset is 24 rooms, with each room having a given capacity. For semester1-2008/09 the dataset contains a total of 165 examination papers across 23 programs offered by 7 faculties. The total number of students is 4,284 with 15,416 enrolments. The conflict density matrix is 0.05. The total available exam space for this dataset is 28 rooms. Table 2 summarises these datasets.

The number of exam days and timeslots are 10 and 20 respectively with 2 timeslots on each examination day. There are no exams during the weekend (Saturday and Sunday), hence we exclude the weekend timeslots in our timeslot indices. The timeslot indices are as follows: *1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23* and *24*. Indices *1* and *2* refer to day 1, timeslot *3* and *4* refer to day 2 and so on. There are no indices *11* to *14* because these indices would refer to Saturday and Sunday.

Table 2.2 Summary of the UMP dataset

Categories	Semester 1-2007/08	Semester 1-2008/09
Exams	157	165
Students	3,550	4,284
Enrolments	12,731	15,416
Conflict density	0.05 (5%)	0.05 (5%)
Timeslots per day	2	2
Rooms	24	28

2.8 Experimental setup

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 semester 1-2007/08 is 4.74 and 20.74, and in semester1-2008/09 it is 6.16 and 23.11 respectively. Hence, the (randomly) selected initial solutions in semester1-2007/08 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. This move is only possible when the destination room is empty.

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. As an example if timeslot 2 and timeslot 6 were selected, move exams in timeslot 2 to timeslot 6 and vice versa.

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

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

2.9 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). We used Delphi programming language. 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 semester1-2007/08 is shown in table 3 and semester1-2008/09 is shown in table 4.

2.9.1 Semester1-2007/08

2.9.1.1 *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, no.1 in table 1) (Kahar and Kendall, 2010a). Table 3 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.

2.9.1.2 *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 *modified*-GDA with 1500 iterations (table 3), 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 is able to produce a better solution but with a smaller 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.

2.9.1.3 *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 iterations producing 4.94 with Nh7. Comparing *modified*-GDA and Dueck-GDA with 1500 iterations (table 3), the *modified*-GDA is 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 is 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 methods describe above is shown in figure 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 (when compared to the constructive heuristic with a candidate list of five).

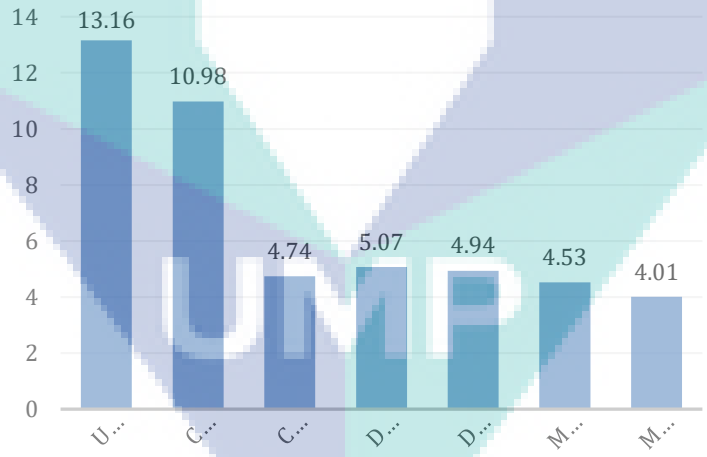

Figure 2.2 Best values of each method for semester1-2007/08

2.9.2 Semester 1-2008/09

2.9.2.1 *Modified*-GDA vs UMP proprietary software

In semester 1-2008/09, 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

Table 2.3 GDA result for semester1-2007/08

Neighbourhood moves	Initial Cost	*Modified*-GDA 480 sec *Ave*	*Stdev*	*Min*	*Max*	*Modified*-GDA 960 sec *Ave*	*Stdev*	*Min*	*Max*	*Dueck*-GDA 480 sec *Ave*	*Stdev*	*Min*	*Max*	*Dueck*-GDA 960 sec *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; var = variance; stdev = standard deviation; min = minimum; max = maximum

Table 2.4 GDA result for semester1-2008/09

Neighbourhood moves	Initial Cost	*Modified*-GDA: 1500 iterations ≈ 8 min				*Modified*-GDA: 3000 iterations ≈ 16 min				*Dueck*-GDA: 1500 iterations ≈ 8 min				*Dueck*-GDA: 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

(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.

2.9.2.2 *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).

2.9.2.3 *Modified*-GDA vs *Dueck*-GDA

For *Dueck*-GDA, with 1500 iterations it is able to produce a solution of 7.20 using Nh9 and with 3000 iterations produces 6.39 with Nh7 (see table 4). Comparing *modified*-GDA and *Dueck*-GDA with 1500 iterations (table 4), the *modified*-GDA is 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 a poorer initial cost of 16.68, the *modified*-GDA is 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 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. In the next section, we will further analyse the results.

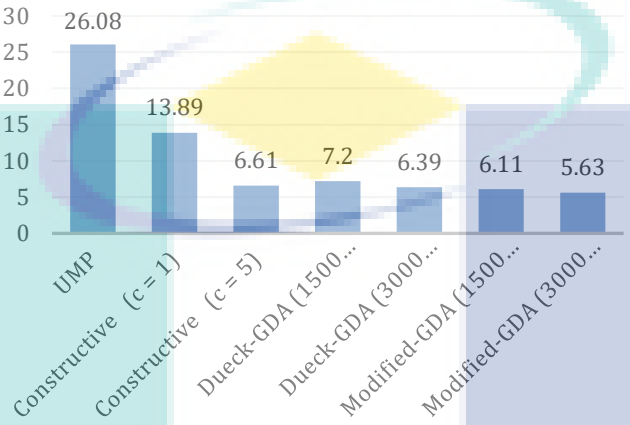

Figure 2.3 Best values of each method for semester1-2008/09

2.10 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 by using different neighbourhood heuristics?

Note that the analyses in (a) to (c) concentrates 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.

The statistical tests are carried out using *t*-test, one-way ANOVA, Games Howell post-hoc, Kruskal-Wallis or 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 t-test and Mann-Whitney U are used to compare two samples while one-way ANOVA and Kruskal-Wallis are used to compare more than two samples. 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 sample simultaneously. Additionally, Mann-Whitney U is used to investigate the rejection cause of H_0 in conjunction with Kruskal-Wallis.

The t-test and one-way ANOVA are used with a normally distributed sample while the Kruskal-Wallis and Mann-Whitney U are used for non-normally distributed sample. 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 then continue with the relevant statistical test (as described above) based on the normality test result.

2.10.1 Semester1-2007/08

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

We analyse the *modified*-GDA and *Dueck*-GDA result using *t*-test and Mann-Whitney U. Table 5 and table 6 show the *p*-value result for 1500 iterations and 3000 iterations respectively. For 1500 iterations, (see table 5), we notice that most of the results show 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), again most of the results 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.

Table 2.5 Semester1-2007/08 *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 *MwU*	.000 *MwU*	.000 *t*	.000 *MwU*
Nh2	.694 *MwU*	.221 *MwU*	1.00 *MwU*	1.00 *MwU*
Nh3	.000 *MwU*	.000 *t*	.000 *MwU*	.000 *MwU*
Nh4	.000 *t*	.000 *t*	.000 *t*	.000 *t*
Nh5	1.00 *MwU*	.385 *MwU*	1.00 *MwU*	1.00 *MwU*
Nh6	.299 *MwU*	.037 *MwU*	.009 *MwU*	.000 *t*
Nh7	.000 *t*	.000 *MwU*	.000 *t*	.000 *MwU*
Nh8	.000 *t*	.000 *t*	.466 *t*	.900 *MwU*
Nh9	.000 *t*	.005 *t*	.000 *t*	.000 *t*
Nh10	.000 *t*	.000 *MwU*	.015 *t*	.251 *MwU*

MwU = Mann-Whitney U; t = t-test

Table 2.6 Semester1-2007/08 *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 *t*	.000	.000 *t*	.000 *t*
Nh2	.019	.427	1.00	1.00
Nh3	.000 *t*	.000 *t*	.000	.000
Nh4	.000 *t*	.000 *t*	.000 *t*	.000 *t*
Nh5	1.00	.688	1.00	1.00
Nh6	.018 *t*	.115	.024	.216 *t*
Nh7	.000 *t*	.000 *t*	.000	.034
Nh8	.000 *t*	.503 *t*	.509 *t*	.296
Nh9	.000 *t*	.000 *t*	.000 *t*	.000 *t*
Nh10	.000 *t*	.375 *t*	.013	.652

MwU = Mann-Whitney U; t = t-test

2.10.1.2 Comparing initial costs

We compare the initial cost based on the number of iterations for all neighbourhood heuristics. We use one-way ANOVA (for normally distributed data) and Kruskal-Wallis (for non-normal data) 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 exists enough evidence to conclude that there is a difference (reject

H_0) among the results produced between the initial costs for all of the neighbourhood heuristics (see table 7). Referring to table 7, the *p*-values are all less than 0.05 which leads us to reject H_0.

In-depth analyses 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 Games Howell Post-Hoc (in conjunction with one-way ANOVA) and Mann-Whitney U (in conjunction with Kruskal-wallis). Based on the analysis only a few of the initial costs show 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. From this analysis we conclude that it is beneficial to start with the best quality solution possible.

Table 2.7 Semester 1-2007/08 *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^{kw}	.000^{owA}
Nh2	.000^{kw}	.000^{kw}
Nh3	.000^{kw}	.000^{kw}
Nh4	.000^{owA}	.000^{owA}
Nh5	.000^{kw}	.000^{kw}
Nh6	.000^{kw}	.000^{kw}
Nh7	.000^{kw}	.000^{kw}
Nh8	.000^{kw}	.000^{owA}
Nh9	.000^{owA}	.000^{owA}
Nh10	.000^{kw}	.000^{kw}

owA = one-way ANOVA; kw = Kruskal-Wallis

Table 2.8 Semester 1-2007/08 *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^{MwU}	.000^{t}	.000^{t}	.000^{MwU}
Nh2	.087^{MwU}	.340^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	.000^{MwU}	.000^{t}	.051^{MwU}	.001^{t}
Nh4	.090^{t}	.141^{t}	.755^{t}	.992^{t}
Nh5	1.00^{MwU}	.038^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	.860^{MwU}	.044^{MwU}	.524^{MwU}	.073^{t}
Nh7	.000^{t}	.000^{MwU}	.456^{t}	.000^{MwU}
Nh8	.273^{t}	.817^{t}	.624^{t}	.589^{MwU}
Nh9	.129^{t}	.002^{t}	.040^{t}	.692^{t}
Nh10	.065^{t}	.303^{MwU}	.005^{MwU}	.172^{MwU}

MwU = Mann-Whitney U; t = t-test

2.10.1.3 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 either t-test or Mann-Whitney U depending on whether the data is normally distributed. Table 8 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 8):

- 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, a diversified 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 3) 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).

2.10.1.4 Comparing neighbourhood heuristics

We compare all the 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 9 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 10 shows a summary of the non significant differences (accept H_0) between the neighbourhood heuristics. Referring to table 10, 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.

Finally, we can summarise that Nh1 produces the best result followed 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 11.

Table 2.9 Semester 1-2007/08 *p*-value comparison for the neighbourhood heuristics based on the initial cost and the number of iterations

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

Table 2.10 Semester 1-2007/08 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,	-	Nh4,	-
Nh4	-	-	Nh7	-	Nh7	-	Nh7	-
Nh5	-	-	-	-	-	-	-	-
Nh6	-	Nh8,	-	-	Nh9	Nh9	-	Nh9
Nh7	-	-	-	-	-	-	-	-
Nh8	-	Nh10	-	Nh10	-	Nh10	Nh10	Nh10
Nh9	-	-	-	-	-	-	-	-
Nh10	-	-	-	-	-	-	-	-

'-' = result show rejecting H_0

2.10.2 Semester1-2008/09

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

As in the previous section (9.11), we used *t*-test and Mann-Whitney U to analyses the result. Table 11 and table 12 show the *p*-value result for 1500 iterations and 3000 iterations respectively. For 1500 iterations, (see table 11), 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 12), 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, semester1-2008/09 dataset show more non-significant

difference compared to semester1-2007/08. However, the result that shows non-significant difference mainly involves neighbourhood heuristic that performs poorly (same as in semester1-2007/08 result).

Table 2.11 Semester1-2008/09 *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^{t}	.000^{MwU}	.000^{t}	.000^{MwU}
Nh2	.080^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	.000^{MwU}	.000^{t}	.000^{MwU}	.000^{MwU}
Nh4	.000^{t}	.000^{t}	.000^{t}	.000^{t}
Nh5	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	.074^{MwU}	.879^{t}	.467^{t}	.000^{MwU}
Nh7	.000^{t}	.000	.000^{MwU}	.000^{t}
Nh8	.000^{t}	.466^{t}	.476^{t}	.734^{MwU}
Nh9	.000^{t}	.000^{t}	.000^{t}	.436^{t}
Nh10	.000^{t}	.003^{t}	.844^{MwU}	.330^{MwU}

MwU = Mann-Whitney U; t = t-test

Table 2.12 Semester1-2008/09 *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^{t}	.000^{t}	.000^{MwU}	.000^{t}
Nh2	.600^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	.000^{t}	.000^{MwU}	.000^{t}	.000^{MwU}
Nh4	.000^{t}	.000^{t}	.000^{MwU}	.000^{t}
Nh5	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	.035^{MwU}	.001^{t}	.463^{t}	.000^{MwU}
Nh7	.000^{t}	.000^{MwU}	.000^{MwU}	.000^{MwU}
Nh8	.000^{t}	.473^{MwU}	.196^{t}	.474^{MwU}
Nh9	.000^{t}	.000^{t}	.164^{t}	.144^{MwU}
Nh10	.000^{t}	.995^{t}	.177^{MwU}	.288^{MwU}

MwU = Mann-Whitney U; t = t-test

2.10.2.2 Comparing initial costs

We compare the initial cost based on the number of iterations for all neighbourhood heuristics for semester1-2008/09 dataset. As in section 9.1.2, we used one-way ANOVA (normally distributed data) and Kruskal-Wallis (non-normal data) to compare between the initial costs (i.e. 18.40, 15.25, 12.30 and 9.21). Referring to table 13, 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 Games Howell Post-Hoc and 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 4, it is best to start with a good quality solution and thus reaffirms our conclusions in section 9.1.2.

Table 2.13 Semester1-2008/09 *p*-value comparison for the initial cost for each neighbourhood heuristic based on the number of iterations

Neighbourhood heuristics	*p-value*	
	1500	3000
Nh1	.000^{kw}	.000^{owA}
Nh2	.000^{kw}	.000^{kw}
Nh3	.000^{kw}	.000^{owA}
Nh4	.000^{owA}	.000^{kw}
Nh5	.000^{kw}	.000^{kw}
Nh6	.000^{kw}	.000^{kw}
Nh7	.000^{kw}	.000^{kw}
Nh8	.000^{kw}	.000^{kw}
Nh9	.000^{owA}	.000^{kw}
Nh10	.000^{kw}	.000^{kw}

owA = one-way ANOVA; kw = Kruskal-Wallis

Table 2.14 Semester1-2008/09 *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^{t}	.000^{t}	.000^{t}	.000^{MwU}
Nh2	.907^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh3	.000^{MwU}	.000^{t}	.001^{t}	.000^{t}
Nh4	.020^{t}	.177^{t}	.124^{MwU}	.811^{t}
Nh5	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}	1.00^{MwU}
Nh6	.000^{MwU}	.094^{t}	.992^{t}	.029^{MwU}
Nh7	.000^{t}	.000^{MwU}	.000^{MwU}	.000^{t}
Nh8	.000^{t}	.000^{MwU}	.742^{t}	.757^{MwU}
Nh9	.035^{t}	.744^{t}	.155^{t}	.322^{MwU}
Nh10	.033^{t}	.084^{t}	.450^{MwU}	.752^{MwU}

MwU = Mann-whitney U; t = t-test

2.10.2.3 Comparing the number of iterations

As in 9.1.3, 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 14 shows the *p*-value of the comparison between the number of iterations. At the 95% confidence interval, the result is as follows (see table 8):

- 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 semester1-2007/08 and this reaffirms our conclusion (as in the previous dataset) that is best to use a larger number of iterations.

2.10.2.4 Comparing neighbourhood heuristics

As in 9.1.4, we compare the set of neighbourhood heuristics based on the initial costs and the number of iterations using Kruskal-Wallis. Table 15 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. Table 16 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.

Table 2.15 Semester1-2008/09 *p*-value comparison for the neighbourhood heuristic based on initial cost and the number of iterations

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

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 performance is different between the two datasets. Hence, a neighbourhood that works for one dataset might not necessarily work on other dataset. Therefore, it is best to use a set of diversified neighbourhood heuristics (e.g. Nh1 and Nh7) as it will encourage exploration of the search space.

Table 2.16 Semester1-2008/09 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,	-	-	-	-
Nh4	-	-	Nh9	Nh9	-	-	Nh9	Nh9
Nh5	-	-	-	-	-	-	-	-
Nh6	Nh8,	-	Nh10	-	Nh8,	-	Nh8	-
Nh7	-	-	-	-	-	-	-	-
Nh8	Nh9	Nh10	Nh10	-	Nh10	Nh9	Nh10	
Nh9	-	-	-	-	-	-	-	-
Nh10	-	-	-	-	-	-	-	-

'-' = result show rejecting H_0

2.11 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 that the *Modified*-GDA is able to produce better quality solutions than *Dueck*-GDA. Some of the neighbourhood heuristics show non-significant difference. However, this 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 semester 1-2007/08 show more similarity compared to semester 1-2008/09. 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 neighbourhood heuristic is unable to diversify the search when it is near to a local optima. Referring to table 3 and table 4, we can summarise that using a smaller initial cost produce a higher quality solution when compared to using a larger initial cost. 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 iterations. We notice that the result is very much dependent on the heuristics used. A diversified 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 table 2). In our observation, Nh1 (which produce the best result) is a robust neighbourhood heuristics (see table 3 and table 4). 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 semester 1-2007/08 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.

2.12 Statement of contribution

This paper has presented a study of a real-world examination timetabling problem from UMP. It involves scheduling exams into timeslots and rooms, and improving upon a constructive heuristic. The contributions of this paper are as follows:

a) We have presented 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.
b) We have presented an implementation of a *modified*-GDA in solving a 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.
c) We have investigated 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.

2.13 Conclusion and future research

In this paper, 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. For future work, our aims are to further explore the *modified*-GDA approach:

- 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.
- Use all of the neighbourhood heuristics in every iteration.
- Combine different neighbourhood heuristics. Example using Nh4 or Nh6 in early stage of the search and then Nh1 or Nh7 in the latter stages. We believe this could save computational time if a suitable neighbourhoods are combined in an intelligent ways.

Acknowledgements

The examination/invigilator dataset has been provided by the Academic Management Office (AMO), UMP and the research has been supported by the Public Services Department of Malaysia (JPA) and the Universiti Malaysia Pahang (UMP).

CHAPTER 3

EVALUATING UMP EXAMINATION TIMETABLE

Abstract

This work presents a study of examination timetabling problem from Universiti Malaysia Pahang (UMP). UMP operated from two campuses (i.e., Gambang and Pekan) and this formed new constraints for consideration in producing quality UMP examination timetable. These new constraints include schedule exams into appropriate campus and similar exams held in different campus must be assigned to the same timeslot. These constraints have not been examined before in the literature. UMP unable to evaluate examination timetable quality due to having no formal mathematical model. Hence, this paper aims to investigate the UMP examination timetabling constraints by developing a formal mathematical model and evaluate the current UMP examination timetable using the proposed formal mathematical model.

Keywords: Computational Intelligence, Examination Timetabling, Scheduling

3.1 Introduction

The examination timetabling problem involves allocating examinations to a fix number of rooms and timeslots whilst fulfilling the constraints. The constraints differ from one institution to another. This constraints can be categorised as hard and soft constraints. The hard constraints must be satisfied. An example, no students were assign two examinations simultaneously. Timetable that meet the hard constraint are considered as a feasible timetable. The soft constraints refer to those requirements that need be met as much as possible[6]. An example, maximise spreading of student examination papers. This would allow student to rest and do revision between their exam papers (which is preferred by many students). Hence, the soft constraints (also referred to as objective function) enable us to determine timetable quality. This involve a mathematical model that calculate the penalty cost value for every soft constraints violation. For a quality examination timetable, the total penalty value need to be minimised (Abdullah, S. and Turabieh, H., 2012; Ayob, M., et. al., 2007).

The uncapacitated problem does not consider room capacity unlike capacitated problem which considers room capacity as one of the hard constraint (Abdullah, 2006; Pillay and Banzhaf, 2009). Capacitated problem resemble the real-world problem because it take into account the room capacity as a hard constraint. The capacitated problems are more complex and challenging to solve compared to the uncapacitated problems. The room constraint increases the level of complexity to the overall problem in producing a high quality examination timetable (Burke, Newall and Weare, 1996b)

In this paper, we present an investigation of Universiti Malaysia Pahang (UMP) examination timetabling problem which consists different constraints from the literature (Kendall and Hussin, 2005a). Related work in examination timetabling is presented in section 2. The UMP examination timetabling problem and its constraints are presented in section 3. In section 4, the proposed UMP examination timetabling formal model is discussed. Section 5 and section 6 desribe the investigated data and discussion on the results respectively. Finally, the conclusion and recommendation for the future work is discussed in section 7.

3.2 Related Work

The most common examination timetabling datasets seen in the literature are the Toronto dataset (Carter, Laporte and Lee, 1996), University of Nottingham dataset (Burke, Newall and Weare, 1996b) and from University of Melbourne (Merlot et. al., 2003). Over the years, many researcher proposed new examination dataset. This include the 2007 Second International Timetabling Competition (refer to as ITC2007) dataset (McCollum et. al. 2008), UiTM dataset (Kendall, and Hussin, 2005a; Kendall and Hussin, 2005b), UKM dataset (Ayob, Abdullah and Malik, 2007) and UMP dataset (Kahar and Kendall, 2010).

The Toronto dataset is an uncapacitated examination timetabling problems from thirteen different academic institutions around the world (Carter, Laporte and Lee, 1996). These datasets differ in term of the number of examinations, timeslots and registered students as well as the conflict density. For example, instance car-s-91 consists of 35 timeslots, 682 examinations, 16925 students and conflict density is 0.13 while instance car-f-92 consists of 32 timeslots, 543 examinations, 18419 students and conflict density is 0.14. The Toronto dataset consider only clash free as the hard constraint and spreading of examinations as the soft constraints. Many researchers have investigated Toronto dataset including (Abdullah et. al., 2010; Alzaqebah and Abdullah, 2014; Burke and Bykov 2012; Sabar et. al., 2012a; Turabieh

and Abdullah, 2011b). The Nottingham dataset was introduced by Burke (Burke, Newall and Weare, 1996b). Nottingham dataset is a capacitated problem which consists of 7896 students, 23265 student enrolments, 1550 room capacities, 23 available timeslot and 0.03 conflict density. The objective of the Nottingham dataset is to reduce the back-to-back exams on the same day (Burke, Newall and Weare, 1996b)

Merlot et. al., (2003) introduced a dataset from University of Melbourne. They presented two capacitated datasets which are from semester I (mel01s1) and II (mel01s2) in 2001. Semester I has 609 exams and 28 available timeslots, while semester II has 657 exams with 31 available timeslots. There are two timeslots on every weekday and different capacity for each timeslot. There are five rooms in which exams can be assigned. The soft constraint is to minimise the back-to-back exams on the same day or overnight.

The second international timetabling competition (ITC2007) dataset was introduce in August 2007. Twelve datasets were introduced with different conflict density, number of examinations, total students, available timeslots and rooms. The ITC2007 examination dataset hard constraints include no clashing, exams capacity should meet the room capacity and the examinations should follow the specified arrangement. The soft constraint of the ITC2007 examination dataset include, minimising back-to-back exams on the same day, timeslot and room usage, mixed examinations length in a timeslot and assign large capacity exam as early as possible in the exam period. Many researchers have investigated ITC2007 examination dataset (Alzaqebah and Abdullah, 2014; Anwar et. al., 2013; Battistutta, Schaerf and Urli, 2015; Bykov and Petrovic, 2016; Gogos, Alefragis and Housos, 2012).

An uncapacitated dataset from UiTM Malaysia was introduced by Kendall and Hussin, (2005a). UiTM dataset consists of 2063 examinations, 84675 registered students, 357761 student enrolments and 40 timeslots. The hard constraints include no clashing. The soft constraint include spreading the exams over the exam period and penalise those exams being scheduled on weekend.

A capacitated examination dataset from UKM, Malaysia was introduced by Ayob et. al., (2007). The dataset contains 818 examinations, 14047 registered students as well as 75857 student enrolments, 42 timeslots and total capacity is 1550. The hard constraints include no clashing and not allowing students to seat three consecutive examinations in the same day. Additionally, students assign to seat consecutive exam must be allocated to the same exam room. They also include avoiding room sharing for some exams. The soft constraint are spreading and minimise back-to-back exams on the same day.

A capacitated dataset from UMP Malaysia was introduced by Kahar and Kendall (2010). Two different datasets from different semester were presented. The first dataset contains 157 examinations, 3550 students, 12731 student enrolments, 24 rooms and 0.05 conflict density. The second dataset contains 165 examinations, 4284 registered students, 15416 student enrolments, 28 rooms and 0.05 conflict density. They proposed two new hard constraints that include exams can be scheduled to multiple room but it must be room in the same building only. Additionally, the distance between the rooms must be as close as possible. Besides, here UMP does not allow multiple examinations sharing the same room (Kahar, and Kendall, 2015; Mandal and Kahar, 2015)

Table 3.1 shows the examination timetable benchmark datasets. Toronto, Nottingham and Melbourne datasets are earlier datasets compared to the ITC2007, UiTM, UKM and UMP datasets.

Table 3.1 Examination timetable benchmark datasets

Datasets	Instances	Exams	Students	Enrolments	Conflict density	Timeslot	Room (capacity)
Toronto	13	80 - 2419	549 - 30029	5689 - 120681	0.03 - 0.42	10 - 42	n/a
Nottingham	1	800	7896	34265	0.03	23	1 (1550)
Melbourne	2	521 - 562	19816 - 20656	60637 - 62248	n/a	28 - 31	5 (3024)
ITC2007	12	78 - 1096	655 - 16439	n/a	0.01 - 0.18	12 - 80	1 - 50
UiTM	1	2063	84675	357761	n/a	40	n/a
UKM	1	818	14047	75857	n/a	42	7 (1550)
UMP	2	157 - 165	3550 – 4284	12731 - 15416	0.05	20	24 - 28

3.3 Examination Timetabling Problem at UMP

The Universiti Malaysia Pahang (UMP) was established on February 16, 2002, and is located in the east coast state of Pahang. In 2007, there are only five faculties with almost 3550 students. In 2014, UMP expanded to nine faculties with a total number of student increases to 7833. UMP currently operated from two different campuses that is located in Gambang and Pekan as the main campus. The distance between the two campuses is about 50km apart. Operating from two separated campuses presents many challenges in satisfying the hard and soft constraints.

In UMP, a proprietary system has been used to produce the examination timetable since year 2003. The proprietary system managed to produce the examination timetable but it is unable to evaluate the timetable quality. This is because the proprietary system has no formal mathematical model to evaluate the examination timetable quality. Hence, this motivates us (apart from the reasons mention above) to develop a formal mathematical model of the UMP examination timetabling problem. Table 3.2 describes the UMP examination timetabling constraints.

Table 3.2 UMP examination constraints

	UMP Examination Constraints
Hard Constraints	H1. No students allowed to sit more than one examination simultaneously.
	H2. The capacity of room must be able to accommodate the total number of students.
	H3. A single exam must be split into rooms in the same building.
	H4. The exam needs to be scheduled to the appropriate campus.
	H5. Similar exams held in different campus must be assigned to the same timeslot.
	H6. Larger examination needs to be scheduled early in the examination period.
Soft onstraints	S1. The exam needs to be spread out evenly throughout the exam period.
	S2. The rooms distance of an exam held in multiple rooms should be as close as possible.
	S3. Minimise splitting of an exam over multiple rooms.

**H4, H5 and H6 is the new constraints*

3.4 Proposed UMP Examination Timetabling Formal Model

This section discusses the UMP examination timetabling constraints and the proposed formal model.

Indices

i, j $1 \ldots N$

r, p $1 \ldots R$

s $1 \ldots S$

t $1 \ldots T$

Parameters

N Number of exams

R Number of rooms

S Number of registered students

T Number of timeslots

S_i Number of registered students for exam i

L_i Campus location for exam i

M_r Campus location for room r

B_r Building for room r

f_r Total capacity for room r

c_{ij} Conflict matrix where each element, c_{ij} represents the number of students registering in both exams i and j.

d_{rp} Distance matrix where each element, d_{rp} represent the distance between rooms r and p.

v_{ij} Coincidence matrix where value 1 in v_{ij} represents the exams that must be schedule in the same timeslot for both exams i and j, 0 otherwise.

Decision variables

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

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

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

The objective function (i.e. soft constraints) include spreading examinations over the exam period, minimise the room distance for an exams assigned to multiple rooms and minimise splitting of an exam over multiple rooms. The formulation is shown as bellow:

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

The first component of the objective function is F_1 (i.e. spreading cost) is formulated 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

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

t_i and t_j represent the timeslots schedule for exam *i* and *j*. Eq.2 represents the spreading cost of exams *i* and exam *j*, and it is calculated by multiplying with total students in conflict c_{ij} with the proximity value[12]. The proximity values used are 16, 8, 4, 2 and 1. Proximity value 16 represent the exams being schedule consecutively; proximity value 8 represent the exams has one timeslot gap, value 4 represent the exams have two timeslot gaps and so on.

The second component of the objective function is the distance cost, F_2 (minimize the distance between rooms for those exam assigned in multiple rooms) is shown in eq.4:

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

The third component of objective function is the splitting cost, F_3 (splitting of an exam over multiple rooms) is formulated in eq.5:

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

Where h_i is the number of rooms exam *i* has been split across. h_i can be calculated using eq.6.

$$h_i = \sum_{r=1}^{R} y_{ir} \; for\ all\ i \in \{1, \dots, N\} \quad \text{(eq.6)}$$

The following is discussion of the hard constraints:

a. Students cannot assigned to sit more than one examination simultaneously (see eq.7). If both exam *i* and exam *j* are scheduled to the same timeslot *t*, the number of students register in both exams *i* and *j* must be equal to zero (i.e. $c_{ij} = 0$). Otherwise the result will return a value other than zero which resulting student clashing examinations.

$$\sum_{t=1}^{T}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} x_{it} x_{jt} c_{ij} = 0 \quad \text{(eq.7)}$$

b. Exams *i* must be assigned only once in timeslot, *t*.

$$\sum_{t=1}^{T} x_{it} = 1 \quad \text{for all } i \in \{1, \dots, N\} \quad \text{(eq.8)}$$

c. Exam *i* must be split into rooms in the same building only. This requirement allow

lecturer to easily assist (i.e. unclear question etc) students during exam.

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

where

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

d. The number of exam rooms used in timeslot t must not exceed the total rooms in a timeslot t, R_t.

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

e. The number of students assigned to a room(s) must be less than the maximum room(s) capacity.

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

f. The exam needs to be scheduled to the appropriate campus.

$$\sum_{r=1}^{R} y_{ir}.campus(M_r, L_i) = 0 \qquad for\ all\ i \in \{1,\dots,N\} \quad \text{(eq. 12)}$$

where

$$campus(M_r, L_i) = \begin{cases} 0 & if\ (M_r = L_i) \\ 1 & otherwise \end{cases}$$

g. Exam with the same code but held in different campus need to be assigned in the same timeslot. This is consider as coincidence constraints and these exam are presented in v_{ij}

$$\sum_{i=1}^{N}\sum_{j=1}^{N-1} v_{ij} \cdot timeslot(t_i, t_j) = 0 \qquad For\ all\ t \in \{1,\dots,T\} \quad \text{(eq.13)}$$

Where

$$timeslot(t_i, t_j) = \begin{cases} 0 & if\ (t_i = t_j) \\ 1 & otherwise \end{cases}$$

h. Larger examination needs to be scheduled early in the examination period. Examination with capacity greater than 400 need to be schedule in timeslot 1 to timeslot 10 only (i.e. first week of the examination period).

$$\sum_{i=1}^{N} exam\_cap(S_i) \cdot tSlot\_max(t_i) = 0 \qquad \text{(eq.14)}$$

Where

$$exam\_cap(S_i) = \begin{cases} 1 & if\ S_i > 400 \\ 0 & otherwise \end{cases}$$

and

$$tSlot\_max(t_i) = \begin{cases} 1 & if\ t_i > 10 \\ 0 & otherwise \end{cases}$$

3.5 Investigated Data

In this section, we describe the examination dataset used to evaluate the formal model proposed in section 4. The experiment dataset used are from semester1-2014/2015 and semester2-2014/2015. In semester1-201415, the total number of exams is 445, number of students is 7,833 and 31,185 number of student enrolments. While in semester2-201415, the total number of exams is 426, number of students is 7,157 and 27,526 number of student enrolments. The total exam room is 40 in both semester. Table 3.3 show the dataset information.

Table 3.3 Summary of UMP investigated datasets

Categories	Semester1-	Semester2-
Exams	445	426
Students	7,833	7,157
Enrolments	31,185	27,526
Conflict	0.04 (4%)	0.04 (4%)
Timeslot per	2	2
Rooms	40	40

The number of examination days is 14 with two timeslots on every examination days. No exam assigned on Saturday and Sunday. Figure 3.1 and 3.2 show the timeslot indices used in semester1-2014 and semester2-201415 respectively. Referring to figure 1 and 2, indices 1 and 2 refer to day 1, while timeslot 3 and 4 refer to day 2 and so on. The indices 11 to 14 and 25 to 28 are invisible because those indices refer to weekend. These indices were consider but cannot be selected so that we could evaluate the actual examination gap (during weekend). The indices 7 and 8 in semester1-21415 are invisible because those indices refer to public holiday and no exam assigned on that day.

(1, 2, 3, 4, 5, 6, 9, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38)

Figure 3.1 Timeslot indices for semester1-201415.

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 29, 30, 31, 32, 33, 34, 35, 36)

Figure 3.2 Timeslot indices for semster2-201415.

Table 3.4 Result of the UMP Examination Timetable calculated using proposed formal model

No	Hard Constraints	Result (sem1-201415)	Result (sem2-201415)
H1	No students are allowed to take more than one examination simultaneously.	Not comply	Not comply
H2	The number of students allocated to an exam room must be less than the maximum capacity of room	Comply	Comply
H3	An exam must be split into rooms in the same building.	Not comply	Not comply
H4	The exam needs to be scheduled to the appropriate campus.	Comply	Comply
H5	Similar exams held in different campus must be assigned to the same timeslot	Comply	Comply
H6	Larger examination needs to be scheduled early in the examination period.	Comply	Comply

3.6 Results and Discussion

The discussion on the results of the UMP examination timetabling problem is presented in this section. Table 4 shows the result of the UMP examination timetable used in semester1-201415 and semester2-201415. Based on the result calculated using the proposed model, it indicates that the examination timetable (i.e. semester1-201415 and semester2-201415) generated by UMP did not comply with some of the hard constraints (see Table 4). The UMP examination timetable did not comply with the not allowing students to sit two examinations simultaneously constraint. Those clashing students had to sit two exams consecutively (held straightway after the first exam finish). According to the students it is tiring and stressful because they have to sit for 6 hour exam consecutively (where each exam duration is 3 hours).The second constraint UMP fail to comply is the exam must be split into rooms in the

same building only constraint. A number of 25 exams in semester1-201415 and 12 exams in semester2-201415 were assigned to rooms in different buildings.

The examination timetable quality is evaluated based on the three objectives (see section 4). In semester1-201415, the total penalty cost is 541.36 with the spreading cost F_1 is 11.30, distance cost F_2 is 529.44 and the splitting cost F_3 is 0.62. For semester2-201415, the total penalty cost is 307.96 with spreading cost F_1 is 13.01, distance cost F_2 is 294.41 and the splitting cost F_3 is 0.54. The distance cost for both semester1-201415 and semester2-201415 are high because some exams were assigned to rooms that is in different building. Based on the proposed formal mathematical model, we are able to evaluate the UMP examination timetable used in semester1-201415 and semester2-201415.

3.7 Conclusion and Future Work

Examination timetabling is an important and challenging task faced by every academic institution. Every academic institution has a different constraints and resources in producing the examination timetabling. Hence, the examination timetable should be generated independently to meet the individual requirements. In this paper, a study of the UMP examination timetabling problem and its additional constraints is presented. The formal mathematical model is presented in this paper. Additionally, the quality of the UMP examination timetable used (i.e. sem1-201415 and sem2-201415) is evaluated. The result shows that the examination timetable did not comply with some of the hard constraints (i.e. infeasible timetable). Additionally, the result also shows a high penalty cost (soft constraints). Able to evaluate the timetable solution assist institution to produce timetable that meet the hard constraints (i.e. feasible solution) and as much as possible complying with the soft constraints (i.e. objective function).

For future work, we will implement meta-heuristic approaches technique such as hill-climbing, great deluge algorithm, genetic algorithm and other approaches in producing quality result using the porposed formal model. We hope to produce better result then the timetable currently being used.

Acknowledgement

The examination dataset has been provided to us by the academic office, UMP and the research has been supported by Universiti Malaysia Pahang (UMP).

CHAPTER 4

CONCLUSION

Examination timetabling is a challenging task faced by every academic institution. Every academic institution has a different constraints and resources in producing the examination timetabling. In this research work, a study of the UMP examination timetabling problem and its additional constraints is presented. The UMP formal mathematical model is presented in this research work taken from two different semester. The quality of the UMP current examination timetable being used is evaluated. The result shows that the examination timetable did not comply with some of the hard constraints (i.e. infeasible timetable). Additionally, the result also shows a high penalty cost (soft constraints). Ablility to evaluate the timetable solution assist institution to produce timetable that meet the hard constraints (i.e. feasible solution) and as much as possible complying with the soft constraints (i.e. objective function).

For future work, we will further explore other meta-heuristic approaches technique in order to produce quality result using the porposed formal model. We hope to produce better result then the timetable currently being used.

REFERENCE

Abdullah S, (2006). Heuristic Approaches for University Timetabling Problems. Ph.D. Thesis, School of Computer Science and Information Technology, University of Nottingham, June 2006.

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.

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.

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.

Abdullah, S. and Turabieh, H. (2012). On the use of multi-neighborhood structures within a Tabu-based memetic approach to university timetabling problems. Information Sciences 191(0), 146-168.

Abdullah, S., Turabieh, H., McCollum, B. and McMullan, P. (2010). A Tabu-Based Memetic Approach for Examination Timetabling Problems. Rough Set and Knowledge Technology 6401. Springer Berlin Heidelberg.

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

Anwar, K., Khader, A. T., Al-Betar, M. A. and Awadallah, M. A. (2013). Harmony Search-based Hyper-heuristic for examination timetabling. 9th IEEE International Colloquium on Signal Processing and its Applications (CSPA), pp. 176-181.

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).

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.

Ayob, M., Ab Malik, A. M., Abdullah, S., Hamdan, A. R., Kendall, G. and Qu, R. (2007). Solving a practical examination timetabling problem: a case study. Computational Science and Its Applications–ICCSA 2007: Springer.

Battistutta, M., Schaerf, A. and Urli, T. (2015). Feature-based tuning of single-stage simulated annealing for examination timetabling. Annals of Operations Research, 1-16.

Burke E K and Bykov Y, (2008). A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems, Proceeding PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, Universit de Montral, Montreal, Canada from 19th August to 22nd August, 2008.

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.

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

Burke E K and Newall J, (2002). Enhancing timetable solutions with local search methods. In Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling

(PATAT 2002), volume 2740 of Lecture Notes in Computer Science, pages 195-206, Gent, Belgium, Aug 21-23 2002. Springer.

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

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

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

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.

Burke E K, Eckersley A, McCollum B, Petrovic and Qu R. (2003). Using Simulated Annealing to Study Behaviour of Various Exam Timetabling Data Sets. Proceedings of Metaheuristic International Conference 2003 (MIC 2003), Kyoto, Japan, Aug 2003.

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.

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.

Burke E K, Newall J, Weare, R F, (1996b). 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, pp. 241-250.

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

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

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

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

Burke, E. K. and Bykov, Y. (2012). The late acceptance hill-climbing heuristic (Technical report). University of Stirling, UK: CSM-192.

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

Bykov, Y. and Petrovic, S. (2016). A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling. Journal of Scheduling, 1-14.

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

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

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.

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

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. 404-407.

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,

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, pp. 492-499.

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, 364-382.

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

Gogos, C., Alefragis, P. and Housos, E. (2012). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. Annals of Operations Research 194(1), 203-221.

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.

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

Kahar, M. N. M, and Kendall, (2015). G. A great deluge algorithm for a real-world examination timetabling problem. Journal of the Operational Research Society 66, no. 1 (2015): 116-133.

Kahar, M.N.M., 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

Kendall G and Hussin M N. (2004). 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.

Kendall, G. and Hussin, N. M. (2005a). An investigation of a tabu-search-based hyper-heuristic for examination timetabling. Multidisciplinary Scheduling: Theory and Applications: Springer.

Kendall, G. and Hussin, N. M. (2005b). A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology. Practice and Theory of Automated Timetabling V: Springer.

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

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

Mandal, A. K and Kahar M. N. M (2015). Solving examination timetabling problem using partial exam assignment with great deluge algorithm. In Computer, Communications, and Control Technology (I4CT), 2015 International Conference on, pp. 530-534. IEEE.

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, pp 424-434

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.

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.

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, pp 3–23.

McMullan P, (2007). An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, Lecture Notes in Computer Science, Springer, Vol 4487, pp538-545, 2007.

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., De Causmaecker, P. (Eds.), Selected Papers from Fourth International Conference on Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science, vol. 2740. Springer-Verlag, pp. 207–231.

Petrovic S and Burke E K, (2004). Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chapter University Timetabling. CRC Press.

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

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

Ross P and Corne D (1995). Comparing Genetic Algorithms, Simulated Annealing, and Stochastic Hillclimbing on Timetabling Problems", Evolutionary Computing 2, Springer-Verlag Lecture Notes in Computer Science, ed T.C.Fogarty

Sabar, N. R., Ayob, M., Kendall, G. and Qu, R. (2012a). A honey-bee mating optimization algorithm for educational timetabling problems. European Journal of Operational Research 216(3), 533-543.

Schaerf A, (1999). A survey of automated timetabling. Artificial Intelligence Review, 13(2), pp 87-127.

Silva D L, 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,

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.

Turabieh, H. and Abdullah, S. (2011b). An integrated hybrid approach to the examination timetabling problem. Omega-International Journal of Management Science 39(6), 598-607.

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