Solving a Practical Examination Timetabling Problem: A Case Study

Masri Ayob[1], Ariff Md Ab Malik[1], Salwani Abdullah[1], Abdul Razak Hamdan[1], Graham Kendall[2], and Rong Qu[2]

[1] Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia
43600 Bangi,Selangor, Malaysia
{masri,salwani,arh}@ftsm.ukm.my, ariff215@salam.uitm.edu.my
[2] ASAP Research Group, School of Computer Science and Information Technology,
The University of Nottingham,Nottingham NG8 1BB, UK
{gxk,rxq}@cs.nott.ac.uk

Abstract. This paper presents a Greedy-Least Saturation Degree (G-LSD) heuristic (which is an adaptation of the least saturation degree heuristic) to solve a real-world examination timetabling problem at the University Kebangsaan, Malaysia. We utilise a new objective function that was proposed in our previous work to evaluate the quality of the timetable. The objective function considers both timeslots and days in assigning exams to timeslots, where higher priority is given to minimise students having consecutive exams on the same day. The objective also tries to spread exams throughout the examination period. This heuristic has the potential to be used for the benchmark examination datasets (e.g. the Carter datasets) as well as other real world problems. Computational results are presented.

Keywords: Timetabling, Heuristic, Graph colouring.

1 Introduction

The examination timetabling problem is characterised by assigning a set of exams into a limited number of timeslots subject to a set of constraints (see [1], [2], [3], [12], [16], and [20]). These constraints are usually classified as hard and soft constraints. The hard constraints must be completely satisfied where solutions that satisfy all the hard constraints are called *feasible*. On the other hand, there might be some requirements that are not essential but should be satisfied as far as possible, which are referred to as soft constraints. Common hard constraints for examination timetabling problem are: (i) no student should be required to sit two exams at the same time and (ii) the scheduled exams must not exceed the room capacity. However, in practical examination timetabling problem, there are many other constraints, which are vary among institutions. Similarly in our dataset, we have some unique hard constraints, which we describe in section 2.

A particularly common soft constraint aims to spread exams as evenly as possible throughout the schedule. Due to the complexity of the problem, it is not usually possible to have solutions that do not violate some of the soft constraints. Indeed, the

O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, LNCS 4707, Part III, pp. 611–624, 2007.

quality of the timetable is usually evaluated based on some cost which measures the violation of the soft constraints. A weighted penalty value is associated with each violation of a soft constraint and the objective is to minimize the total penalty value.

Various approaches have been developed for solving examination timetabling problems. These include graph-based heuristics [3], tabu search [4], evolutionary algorithms ([5], [6], [13], [14], and [15]), simulated annealing [7], hyper-heuristics [8], and ant algorithms [9]. [8], for example, employed a tabu search approach to search a sequence of graph heuristics in constructing examination timetables within a hyper-heuristic framework. A move in tabu search chooses a new heuristic list by randomly swapping two graph heuristics in the previous heuristic list. The newly visited heuristic lists are added to the tabu list to avoid re-visiting them. To reduce the computational time, [8] introduced a *failed list* that stored any heuristic list that could not generate a feasible solution. Results showed that the approach can produce good quality solutions that are within the range of the best results reported in the literature for both exam and course timetabling problems.

Graph based heuristics were among the earliest approaches to be used for the timetabling problem ([10] and [3]). They are used as constructive heuristics, constructing solutions by ordering the exams that have not yet been scheduled, according to the perceived difficulty in scheduling that exam into a feasible timeslot. The difficulty of an exam can be represented in various ways such as the degree of conflict it has with other exams, the number of student enrollments etc. Some common graph based heuristics are:

1) **Largest degree first.** This heuristic first schedules the exam that has the largest number of conflicts with other exams.
2) **Colour degree.** Exams with a greater number of conflicts with the exams that have already been scheduled have a higher priority of being scheduled next.
3) **Least Saturation degree.** Exams with fewer feasible slots are scheduled as early as possible. The priority of exams (in the unscheduled list) to be scheduled are changed dynamically during the construction of solution.
4) **Largest weighted degree.** Exams with the higher number of students in conflict are scheduled earlier.
5) **Largest enrolment degree.** Exams with greater number of students are scheduled earlier.

Interested readers are referred to [11], [12], [13], [14], [15], [16], [17], [18], and [19] for more information about examination timetabling.

In this paper, we present a Greedy-Least Saturation Degree (G-LSD) heuristic in order to produce good quality solutions to the real-world examination timetabling problem faced by the University Kebangsaan Malaysia (UKM). The heuristic is an adaptation of the least saturation degree heuristic. We employ a hierarchical problem solving approach to generate solutions for UKM. We utilise a new objective function in order to evaluate the quality of a timetable. The objective function considers both timeslots and days in assigning exams to timeslots. Higher priority is given to minimise students having consecutive exams in the same day. The new objective function differs from the standard proximity cost ([20]), where the only measure is how close the exams are, with no account being taken of exams spanning days (i.e. two consecutive exams, on the same day, are penalised the same as an exam in the

evening and an exam the following morning). The new objective function, and the G-LSD heuristic, can be applied to the standard benchmark examination datasets ([20]) by adding a new variable *day* to each timeslot.

2 Problem Definition

Until recently, human schedulers at the Universiti Kebangsaan Malaysia (UKM) were human decision makers who applied a greedy assignment procedure, based on their experience with a little guidance from computer software to avoid clashes. They did not consider students sitting two/three consecutive exams a day. They would like to take into account spreading exams evenly, and fairly, throughout the timetable but the size/complexity of the problem makes this unrealistic. Therefore, the human scheduler only concerns themselves with the constraint of not assigning a student to sit more than one exam in a given timeslot. Even this procedure usually takes the manual schedulers more than two weeks. After circulating the exam timetable to students, the schedulers invariably receive many complaints from students/lecturers. When students complain about sitting three consecutive exams in a day, they are rescheduled, with an emphasis on making as few adjustments to the rest of the timetable as possible. Usually, if this happen (i.e. complaints about three consecutive exams in a day), a new timeslot may be added (normally on Saturday) and the middle exam is scheduled to this new timeslot. This incurs extra overhead costs. Therefore, this work is motivated by attempting to automate this process, as much as possible, as well as take into account more of the requirements. Indeed, the authors based at UKM have been tasked to this do by the management at UKM.

In this paper, we present our approach and experience in solving the examination timetabling problem for Semester I, year 2006 at UKM. The dataset (UKM06-1) has been preprocessed based on the supplied data which contains 818 exams, 14,047 students, 75,857 enrollments, 42 timeslots and 15 exam days (this excludes the weekend break). The UKM06-1 dataset is held in four text files: UKM06-1.stu, UKM06-1.slt, UKM06-1.rom and UKM06-1.isl, which represent student enrollment, slot, room and isolated exams definition, respectively. The files are available at `http://www.ftsm.ukm.my/jabatan/tk/masri/Exam/`. The exam timetabling problems in UKM (particularly) have different datasets for each semester. In most cases, each student registers for different set of courses. That is the exam timetable of each semester is only valid for that semester. Therefore, in practice, the exam timetable need be generated at the end of each semester.

The dataset has 3 weeks examination period. Each week has 5 exam days (Monday to Friday). Each day has 3 timeslots (morning, afternoon and evening), except Friday which has 2 timeslots (morning and evening only). In order to model the real-world timeslots (in days), we present the vector of days in Fig. 1.

(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 8, 8, 8, 9, 9, 9, 10, 10, 10, 11, 11, 11, 12, 12, 15, 15, 15, 16, 16, 16, 17, 17, 17, 18, 18, 18, 19, 19)

Fig. 1. Vector of days

In Fig. 1, we can see that in most cases we have three entries for each day (e.g. day 1, we have three '1') except on Friday i.e. on day 5, 12 and 19; there are only two *5* entries (representing two timeslots on Friday). Saturdays (days 6 and 13) and Sundays (day 7 and 14) are missing because there are no examinations on Saturday and Sunday. A corresponding timeslot vector is presented in Fig. 2.

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57)

Fig. 2. Vector of timeslots

Fig. 2 shows that the timeslots are represented as indexes. Timeslots 1, 2 and 3 refer to day 1, timeslots 4, 5 and 6 refer to day 2, etc. Note that on day 5 (Friday the first week) there are only 2 timeslots i.e. 13 and 15 (morning and evening sessions). Since there is no afternoon session, we do not use a timeslot with an index of 14. The reason is that we want to utilise different weights since the students have gaps (2 gaps in this case i.e. 15-13=2) even though the exams are scheduled on the same day. Also notice that the timeslot indexes for Saturday the first week (16, 17 and 18) and Sunday the first week (19, 20, and 21) are missing because there are no exams scheduled on these days. The same representation is used for the second and third weeks of the exam period.

The input for the examination timetabling problem can be stated as follows:

- N is the number of exams;
- E_i is the i^{th} exam where $i \in \{1,....,N\}$;
- e_i is number of students sitting exam E_i where $i \in \{1,....,N\}$;
- B is the set of all N exams, $B=\{E_1,..., E_N\}$;
- D is the number of days;
- T is the given number of timeslots (including missing timeslots);
- M is the number of students;
- t_i specifies the assigned time slot for exam E_i, where $t_i \in \{1,..,T\}$ and $i \in \{1,..,N\}$;
- d_i specifies the assigned day for exam E_i, where $d_i \in \{1,..,D\}$ and $i \in \{1,..,N\}$;
- $C=(c_{ij})_{NxN}$ is the conflict matrix where each element denoted by c_{ij}, $(i,j \in \{1,..,N\})$ is the number of students taking both exams E_i and E_j where $c_{ij}=0$ for $i=j$;
- $\Delta t = |t_i - t_j|$ is the timeslot different between exam E_i and E_j;
- $\Delta d = |d_i - d_j|$ is the day different between exam E_i and E_j;
- β_{ij} is a decision variable where $\beta_{ij}=1$ if $c_{ij}>0$ and $i \neq j$; or 0 otherwise.

The constraints of our examination timetabling problem (excluding constraints for assigning exams to rooms) are:

1) All exams must be scheduled and each exam must be scheduled only once.

$$\sum_{s=1}^{T} \lambda_{is} = 1 \qquad \text{for all } i \in \{1,..,N\}. \tag{1}$$

where

$$\lambda_{is} = \begin{cases} 1 & \text{if exam } E_i \text{ is assigned to timeslot } s; \\ 0 & \text{otherwise;} \end{cases} \quad (2)$$

2) No student can sit two exams concurrently. If exam E_i and E_j are scheduled in slot s, the number of students sitting both exams E_i and E_j must be equal to zero, i.e. $c_{ij}=0$.

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} . x(t_i, t_j) = 0. \quad (3)$$

where

$$x(t_i, t_j) = \begin{cases} 1 & \text{if } t_i = t_i; \\ 0 & \text{otherwise;} \end{cases} \quad (4)$$

3) For each timeslot t, the number of students sitting for exams ($Students_t$) must not exceed the maximum number of seats ($MaxSeats$) i.e. 2,400 seats per timeslot for the UKM06-1 dataset.

$$Students_t \leq MaxSeats \qquad \text{for } t \in \{1,..,T\}; \quad (5)$$

4) No student can sit 3 consecutive exams in a day.

$$\text{If } c_{ij} \neq 0;\ c_{ik} \neq 0;\ t_i = x;\ i \neq j \neq k;\ [t_j = x+1 \ \textbf{OR} \ t_j = x-1] \text{ and } d_i = d_j; \\ \text{then } d_k \neq d_i; \quad \text{for all } i,j \in \{1,...,N\}; \quad (6)$$

Due to the complexity of the problem, we have partitioned the problem into two sub problems. That is, assigning exams to timeslots (the same as the capacitated examination timetabling problem but with extra hard constraints, i.e. we impose constraint 4) and the room assignment problem. In this paper, we only consider constraints related to the capacitated examination timetabling problem. Constraints for the room assignment problem will be the subject of our future work.

Before assigning exams to timeslots the supplied data requires some preprocessing. Firstly, we exclude courses with no exams and combine exams that have to be scheduled together into a single exam. We also exclude courses from the Law faculty. These only have two timeslots per day, as their exam period is longer than the other exams. We also exclude co-curriculum courses, which have different (shorter) timeslots. In this work, we use a hierarchical problem solving approach, which focuses on the first stage that is assigning exams to timeslots.

The output of the first stage will be used as input to subsequent stages. The next two stages, which assign exams to (suitably sized) rooms and schedules Law exams into slots and rooms, are not considered here and will be the focus of future work. Since the co-curriculum courses are university level courses which have many student

enrollments, from different faculties, and have shorter exam periods, i.e. different timeslots, they are usually scheduled outside the examination weeks. Therefore, scheduling the co-curriculum exams can be solved independently.

To indicate the density of the conflicting exams, we use *Conflict Density* as defined in [20] and many other works. The *Conflict Density* can be calculated as:

$$\textbf{\textit{Conflict Density}} \quad = \quad \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(\frac{\beta_{ij}}{N}\right)}{N} \tag{7}$$

Higher value of *Conflict Density* shows that, on average many exams are in conflict with each other.

3 Objective Function

As in the standard benchmark datasets, wherever possible, examinations should be spread out over timeslots so that students have large gaps between exams. In order to adhere with practical (real world) issues, we introduce our new cost function, which is adapted from the proximity cost ([20] and the objective function proposed by [13] and [14]). Our *Penalty Cost*, F, is calculated as follows:

$$\textbf{Minimise} \quad F \quad = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} . Penalty \quad (t_i, t_j)}{M} \tag{8}$$

where

$$penalty \quad (t_i, t_j) = \begin{cases} 2^{(5-\Delta t)(2-\Delta d)} \\ \\ 0 \end{cases} \tag{9}$$

Equation 9 presents a weighted *penalty* value that reflects the cost of assigning exam E_i and E_j to a timeslot. These being 0, 1, 2, 4, 8, 16, 64 and 256 where the cost is zero if the gap of timeslot for exam E_i and E_j is greater than 5 or the day gap is greater than 2. We only penalise up to a maximum of 5 timeslots in order to adhere to the well established proximity cost proposed by [20]. We also limit the penalty up to a maximum of 2 days because a 2 day gap between examinations gives enough free time for students. This observation is based on a pilot survey we have carried out at UKM.

The new objective function (equations 8 and 9) aims to minimise the number of students having two exams in a row on the same day (highest penalty) and on the next day (lower penalty); and attempts to spread out exams over timeslots (lowest penalty). Indeed, these formulations emphasises avoiding students having two consecutive exams on the same day instead of avoiding having two consecutive exams on different days. The penalty for students having two consecutive exams on the same

day (*penalty=256*) is higher than the penalty value for students having two consecutive exams on different days (*penalty=16*). This factor is not highlighted in the objective function proposed by [13], [14], and [20]. In fact, [20] ignores the day effect by assuming that the practical time gap between each consecutive timeslot is the same. The objective function introduced in [13] penalised consecutive exams on the same day. That is, [13] only minimised the number of students having two consecutive exams on the same day without spreading exams over timeslots. Subsequently, [14] enhanced the objective function that was proposed in [13] by giving a higher penalty (3) to students having two consecutive exams in the same day and a lower penalty (1) for two consecutive exams spread across two consecutive days. By adapting the three objective functions, we proposed a *Penalty Cost* that combines the features of the above functions into one.

4 Heuristic for the Examination Timetabling Problem

This work presents a heuristic procedure which we call Greedy Least Saturation Degree (G-LSD). We use the term "greedy" because our heuristic attempts to assign each exam to the best timeslot which satisfies all the hard constraints. We randomly assign exams to timeslots when the exam has no conflict with the exams that have already been scheduled. While assigning exams to timeslots, we also ensure that all hard constraints are satisfied. This is an adaptation of the least saturation degree heuristic for classical graph colouring problem. Fig. 3 presents the pseudo-code for the G-LSD. The idea is to give a higher priority to exams still to be scheduled that have the least number of available slots, where the priority changes dynamically during the scheduling process.

In the initialisation step, all exams in B are reset. That is, the number of available timeslots for each exam is set to the maximum available slot and the exam's status is changed to unscheduled and copied into the unscheduled exam set B'={E_1', E_2'….,E_N'}. The heuristic first arranges the unscheduled exams in B′ in non-decreasing order of the number of available timeslots, then in non-increasing order of the number of conflicts they have with other exams (in B) and, finally, by non-increasing order of the number of student enrollments. The heuristic chooses the first exam in B', E_i', and assigns it to the best timeslot that minimises F (equation 8) and subject to the total number of students assigned to the timeslot that does not exceed the maximum seat capacity (i.e. 2400 for UKM06-1 dataset). However, when all slots are available for E_i' (i.e. the exam has no conflict with the exams that have already been scheduled), we randomly choose a timeslot for E_i'. The idea is to allocate the best timeslot for E_i', so as to obtain different solution for each run. While assigning exams to timeslots, we also ensure a clash free schedule (i.e. constraints 1 to 3 are satisfied) and larger exam (student enrollment>=400) are assigned to earlier timeslots (the first two weeks if possible). This is done as, based on discussions with UKM registry officers, they usually assign larger exams to earlier timeslots in order to give longer time for marking larger exams. After assigning exam E_i' to the timeslot, we update the appropriate exam details such as timeslot index, number of available timeslots etc. in B and reduce the number of available timeslots for exams in B' accordingly. Then, we eliminate E_i' from B' and repeat step 2.1 to 2.6 until all the exams have been scheduled, or until E_i' cannot be assigned to any available timeslot.

```
Step 1: Initialisaon
         Initialise all exams in B.
         Copy all exams in B into unscheduled exam set,
         B'={E1', E2'….,EN'}.

Step 2: Assigning Exam to Slot.
         While(B'≠∅)
         {
             2.1   Arrange the exam in B' using Procedure A.
             2.2   Choose the first exam, Ei' in B'.
             2.3   If all timeslots are available for Ei';
                     Then, Randomly choose available timeslot, ti,
                     s.t. the limit of seat capacity of the
                     timeslot and larger size exam should be
                     assigned to earlier timeslot.
             2.4   Else, choose timeslot, ti which has the minimum
                     F, s.t. the limit of seat capacity of the
                     timeslot and larger size exam should be
                     assigned to earlier timeslot.
                     If there is no clash free timeslot for Ei',
                     exit while loop and repeat step 1.
             2.5   Update appropriate exams in B and reduce the
                     number of available timeslots for exams in B'
                     accordingly.
             2.6   Remove Ei' from B'.
         }end while.

Step 3: Verification
         If the solution is feasible concerning constraint (4),
         return the solution.
         Otherwise, repeat Step 1 to Step 3 until the feasible
         solution is obtained or exceed the time limit.
```

Note: Procedure A arrange unscheduled exams in B' in non-decreasing order of number of available timeslots then with non-increasing order of number of conflicts they have with other exams (in B) and non-increasing order of number of students enrollment.

Fig. 3. Psuedo-code of Greedy-Least Saturation Degree (G-LSD) heuristic

If this occurs, we stop the process and start again (steps 1 to 3). After assigning all exams to timeslots, we verify that all hard constraints (constraint 1 to 4) are satisfied. If the solution is feasible, we end the process. Otherwise, we begin again (steps 1 to 3).

Before assigning exam E_i' to a timeslot we ensure that constraints 1 to 3 are satisfied. However, we can only verify if constraint 4 is satisfied when all exams have been assigned to timeslots. This is because we want to reduce the computational time due to the size of problem and the complexity of computing the three consecutive exams in a day. Moreover, our new objective function is able to avoid assigning students sitting three consecutive exams in a day by giving a very high penalty (i.e. *penalty=256*256*) for this case. Therefore, in many cases, our algorithm produces solutions where no students are sitting three consecutive exams in a day. However, the solution is rejected (infeasible) if there are students sitting three consecutive

exams in a day. Since this is a constructive heuristic, we stop the process when we obtain a feasible solution. Of course, we could extend this procedure to produce many feasible solutions and return the best solution found by repeating step 1 to 3 for a given number of iterations.

5 Results

Our heuristic was implemented using Microsoft Visual C++ 6.0. We ran the experiments on an Athlon 1.47 GHz processor, with 512 MB RAM and Windows XP Professional. The G-LSD heuristic was tested on the real world UKM06-1 dataset and also on the benchmark Carter datasets. In order to transform the Carter datasets into a form that is suitable for use with the recently proposed objective function, we apply a day and timeslots vectors (as in UKM06-1; see Fig. 1 and Fig. 2), except that for benchmark Carter datasets we have three timeslots on Friday and one timeslot on Saturday (as suggested in [14] and [15]). We also introduce a six timeslot gap between Saturday morning and Monday morning to cater for the weekend break. For each test, we perform 100 runs. Since there is a random element in selecting each timeslot, when all timeslots are available, we can usually obtain different solutions for each run. In all tests, we use the new *Penalty Cost* (see section 3) function to evaluate the quality of timetable.

Table 1 shows the characteristics of original UKM06-1 and Carter datasets. For capacitated examination timetabling problems, no justification was stated on how the seat capacity limits were set (see [13], [14], and [15]). For example, the seat capacity limit for Car-s-91 was set to 1550 seats. If we divide the number of enrollment by the seat capacity, we need at least 37 timeslots (in ideal case where we can easily assign exams to any slots, i.e. exams have no conflict with each other) such that all students can have a seat in the exam room. We assume this is why [13] increased the number of timeslots from 35 to 51 for Car-s-91 and 23 timeslots to 35 for Tre-s-92. This work also introduces the sta-f-83 instance as a capacitated problem. Later (test B) we run a series of experiments to establish the number of timeslots and the seating capacity required for this revised dataset instance.

Table 1. Characteristics of benchmark and UKM06-1 examination problems

Dataset	Exams	Times-lots	Days	Seat capacity	Students	Enrollment	Conflict density
UKM06-1	818	42	15	2400	14047	75857	0.05
Car-f-92	543	32	12	2000	18419	55522	0.14
Car-s-91	682	35/51	13/19	1550	16925	56877	0.13
Kfu-s-93	461	20	8	1955	5349	25113	0.06
Tre-s-92	261	23/35	9/13	655	4360	14901	0.18
Sta-f-83	139	13	5	-	611	5751	0.14

*Note: * Previously, days have not been properly defined. Based on discussion in [14] and [15], we compute the number of examination days (excluding day off).*
***Seat capacity was introduced later by [13] for five Carter datasets.*

For UKM06-1, we ran an initial test (we call it test A) to establish the best number of timeslots for this dataset. We varied the number of timeslots whilst fixing the

maximum room capacity (i.e. 2,400 seats as currently used). In practice, the number of timeslots available is 42. Actually, due to the difficulties of scheduling large exams, manual schedulers usually allocate certain timeslots for some exams. For example, exams ZZZT1032 and ZZZT1042 which have 2146 and 2110 student enrollments, respectively, were allocated to one entire timeslot for each exam. As a result no other exams were scheduled in these timeslots. Indeed, these exams were enrolled by many students from various faculties, i.e. the conflict density for these exams is very high. However, by using our approach, these exams can be scheduled together with other exams. Although, by setting the number of timeslots is equal to 42 and the maximum room capacity is equal to 2400, human schedulers still struggle to provide good quality timetables. Indeed, they have no clear idea what is the criterion for a good quality timetable and how they can manage to find one, even manually. They always receive many complains after the tentative exam timetable has been circulated to students/lecturers. Therefore, we design test A to show that they can reduce the number of timeslot (that will ultimately reduce the management cost) without scarifying too much on solution quality. Results of this test, is shown in Table 2.

Table 2. Results for the test A on UKM06-1 when varying the number of timeslots

Time-slots	Exam Days	Seat capacity	Number of clash free solutions	Feasible solutions (%)	Average feasible solution, F_{avg}	Best feasible solution, F_{min}
42	15	2400	100	79	4.76	4.25
41	15	2400	100	70	5.30	4.81
40	14	2400	100	35	6.08	5.44
40	14	2106	0	-	-	-
40	14	2250	100	36	6.21	5.86

In Table 2, *'Number of clash free solutions'* indicates the total number of solutions obtained (ignoring violations of students sitting three consecutive exams in a day) over the 100 runs; '*Feasible solutions*' indicates the number of solutions obtained that satisfy all the hard constraints (1 to 4), including no student sitting three consecutive exams in a day. Results in Table 2 shows that we obtained 79%, 70% and 35% feasible solutions when seat capacity=2400 and timeslots is equal to 42, 41 or 40 respectively.

Some secondary rooms (i.e. LobiA(DECTAR) and LobiB(DECTAR)) allocated to the UKM06-1 dataset are actually not preferable due to practical constraints, so we eliminate these rooms from the list of available rooms. For UKM dataset, we also eliminate PSeni(DECTAR) from the list because the room is reserved for Law examinations. Therefore, the total room capacity is reduced to 2340 seats. As a result, we set 2106 or 2250 as the maximum seat capacity limit, i.e. 90% or 96% of the total seat capacity respectively. Table 2 shows that, out of 100 runs, no feasible solution is obtained when the number of timeslots is equal to 40 and seating capacity is equal to 2106. Whereas, we obtained 36 feasible solutions with the best *Penalty Cost,* F_{min} = 5.86 when we used timeslots is equal to 40 and seat capacity is equal to 2250. These results show that our G-LSD is capable of reducing the number of timeslots and avoiding students sitting three consecutive exams a day, as well as spreading out

exams over timeslots. It shows that we can save two timeslots compared to manual scheduler (this might be of interest to the administrator because they could reduce their administration cost). Since the number of timeslots was reduced, the UKM06-1 dataset is now even more constrained.

We carried out a further test (test B) to find out a reasonable number of timeslots required in order to generate feasible solutions (that satisfies all constraints 1 to 4) for the sta-f-83 problem. Initially we set the seat capacity to 2400 such that all students (clash free schedule) have a seat in the exam room. We then, reduced the maximum seating capacity limit to a reasonable value. We also varied the number of timeslots for sta-f-83 starting from 13 timeslots until we found feasible solutions (20 timeslots). Table 3 shows the results of this test. We also extended our experiment to Kfu-s-93, Car-f-92, Car-s-91 and Tre-s-92. We initially tried not changing the number of timeslots and the maximum seat capacity that were set by [20], [13], and [14]. As we are enforcing an additional hard constraint (i.e. avoiding students sitting three consecutive exams in a day), we thought it is worth carrying out some further analysis on these datasets. Table 3 also shows the results of the experiments.

Table 3. Results of test B on Sta-f-83 when varying the number of timeslots

Dataset	Time-slots	Exam Days	Seat capacity	Number of clash free solutions	Feasible solutions	Average feasible solution, F_{avg}	Best feasible solution, F_{min}
Sta-f-83	19	7	2400	100($\mathbf{F'_{min}}$=241.31)	0	-	-
Sta-f-83	20	8	2400	100($\mathbf{F'_{min}}$=189.13)	46	215.58	189.13
Sta-f-83	20	8	428	100($\mathbf{F'_{min}}$=224.41)	18	258.05	241.52
Kfu-s-93	20	8	1955	54($\mathbf{F'_{min}}$**=57.37**)	0	-	-
Kfu-s-93	20	8	2400	67($\mathbf{F'_{min}}$**=55.48**)	0	-	-
Kfu-s-93	40	15	2250	100($\mathbf{F'_{min}}$**=5.86**)	36	6.21	5.86
Car-f-92	49	19	2400	100($\mathbf{F'_{min}}$**=3.02**)	0	-	-
Car-f-92	51	19	2400	100($\mathbf{F'_{min}}$**=2.76**)	10	2.87	2.76
Car-s-91	51	19	2400	100($\mathbf{F'_{min}}$**=4.11**)	0	-	-
Car-s-91	53	20	2400	100($\mathbf{F'_{min}}$**=3.54**)	16	3.98	3.84
Tre-s-92	35	13	655	100($\mathbf{F'_{min}}$**=8.18**)	11	8.88	8.55

Note: The exams day excluded day off (i.e. weekend breaks or public holidays)

Table 4. New characteristics of existing benchmark and UKM06-1 examination datasets

Dataset	Exams	Timeslots	Days	Seat capacity	Students	Enrollment	Conflict density
UKM06-1	818	40	14	2250	14047	75857	0.05
Car-f-92	543	51	19	2400	18419	55522	0.14
Car-s-91	682	53	20	2400	16925	56877	0.13
Kfu-s-93	461	40	15	2250	5349	25113	0.06
Tre-s-92	261	35	13	655	4360	14901	0.18
Sta-f-83	139	20	8	428	611	5751	0.14

Note: We set the maximum seat capacity for sta-f-83 as 428 i.e. 70% of the sum of students.

As a result of the experiments reported above, we propose revised characteristics of the capacitated examination datasets (see Table 4) with the added hard constraint, new *Penalty Cost* and day vector.

6 Conclusion and Future Work

This paper has presented our experience in solving a real-world examination timetabling problem at the University Kebangsaan Malaysia (UKM) with an objective to minimise the *Penalty Cost* using a new objective function. The new objective function attempts to minimise the number of students having two consecutive exams on the same day (highest penalty) and on the next day (lower penalty). It also attempts to spread out exams over timeslots (lowest penalty). Due to the complexity of the problem, we divided the problem into two sub problems; i.e. capacitated examination timetabling and room assignment problems. We employed a hierarchical problem solving approach that has three stages to solve the problem. This work focused on the first stage, that is to assign exams to timeslots subject to a clash free schedule, no student sitting three consecutive exams in a day and total students sitting exams per slot should not exceed the seating capacity. The output of the first stage serves as input for the subsequent stages. The next two stages, which assign exams to rooms and scheduled law exams to slots and rooms will be discussed in our future work.

The Greedy Least Saturation Degree (G-LSD), presented in this work, is an adaptation of basic least saturation degree heuristic for the classical graph colouring problem. As in the standard least saturation degree heuristic, the idea is to give high priority to the exams that are still to be scheduled and which have the least available slots, where the priority changes dynamically during the scheduling process. G-LSD was tested on the UKM06-1 dataset using the new *Penalty Cost* as an evaluation function. Since this is a new *Penalty Cost*, we are not able to make a comparative study. Moreover, no previous work has attempted to impose the hard constraint of no student sitting three consecutive exams in a day. Indeed, it is not fair to compare our solution with manually generated solution since their solution was generated without considering three consecutive exams in a day and not attempted to spread exams over timeslot (i.e. they do not have an objective function).

In order to adhere with practical (real world) constraints, and to further test our approach, we add a new hard constraint to the Carter benchmark exam timetabling dataset in [20], i.e. no student should sit three consecutive exams in a day. That is, we proposed revised characteristics of the capacitated examination datasets with the added hard constraint, new *Penalty Cost* and day vector. This can further validate the approach we have suggested here.

Since this approach has been adapted from a graph colouring approach, it could be applied to other wide range of problems such as course timetabling and school timetabling which can be modeled as graph colouring problems.

We are currently designing and implementing a constructive heuristic for assigning exams to rooms. In order to solve the room assignment problem, we have to model the room assignment problem and design a new objective function because currently, no previous researchers have defined the objective function for room assignment. Our future work will also concentrate on scheduling examinations for a Law faculty which is slightly different from other faculties at the University Kebangsaan Malaysia. This

faculty only has two slots per day (because the exam periods are longer than other exams i.e. they start at 8:30am and 2:30pm). We will also try to schedule invigilators (which is carried manually at the moment) and to incorporate this requirement into the scheduling system.

Acknowledgment. This work was supported by The University Kebangsaan Malaysia (UKM). We wish to thank Nor Ashikin Baharom from Computing Center, UKM and Student Registry Department for fruitful discussion and contribution of UKM dataset.

References

1. McCollum, B.: University timetabling: Bridging the gap between research and practice. In: Burke, E.K., Rudová, H. (eds.): Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling. 30th August-1st September 2006, Brno, Czech Republic, pp. 15–35 (2006)
2. Burke, E.K., Elliman, D.G., Ford, P.H., Weare, R.F.: Examination timetabling in British universities - A survey. In: Burke, E.K., Ross, P. (eds.) Practice and Theory of Automated Timetabling. LNCS, vol. 1153, pp. 76–92. Springer, Heidelberg (1996)
3. Burke, E.K., Kingston, J., de Werra, D.: In: Gross, J., Yellen, J. (eds.) Applications to timetabling. Handbook of Graph Theory, pp. 445–474. Chapman Hall/CRC Press (2004)
4. Di Gaspero, L., Schaerf, A.: Tabu search techniques for examination timetabling. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 104–117. Springer, Heidelberg (2001)
5. Erben, W.: A grouping genetic algorithm for graph coloring and exam timetabling. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 132–158. Springer, Heidelberg (2001)
6. Côté, P.: A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 294–312. Springer, Heidelberg (2005)
7. Dowsland, K.: Off the peg or made to measure. In: Burke, E.K., Carter, M. (eds.) PATAT 1997. LNCS, vol. 1408, pp. 37–52. Springer, Heidelberg (1998)
8. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research 176, 177–192 (2007)
9. Eley, M.: Ant algorithm for the exam timetabling problem. In: Burke, E.K., Rudová, E.K. (eds.) Practice and Theory of Automated Timetabling. LNCS, vol. 1153, pp. 167–180. Springer, Heidelberg (1996)
10. Welsh, D.J.A., Powell, M.B.: The upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal 11, 41–47 (1967)
11. Abdullah, S., Ahmadi, S., Burke, E.K., Dror, M.: Investigating Ahuja-Orlin's Large Neighbourhood Search Approach for Examination Timetabling. OR Spectrum 29(2), 351–372 (2007)
12. Qu, R, Burke, E., McCollum, B., Merlot, L.T.G., Lee, S.Y.: A survey of search methodologies and automated approaches for examination timetabling. Technical Report No. NOTTCS-TR-2006-4, School of Computer Science & IT, University of Nottingham (2006)

13. Burke, E.K., Newall, J.P., Weare, R.F.: A Memetic Algorithm for University Exam Timetabling. In: Burke, E.K., Ross, P. (eds.) Practice and Theory of Automated Timetabling I. LNCS, vol. 1153, pp. 3–21. Springer, Heidelberg (1996)
14. Burke, E.K., Newall, J.P., Weare, R.F.: Initialization strategies and diversity in evolutionary timetabling. Evolutionary Computation 6(1), 81–103 (1998)
15. Burke, E.K., Newall, J.P.: A multistage evolutionary algorithm for the timetable problem. IEEE Trans. Evolutionary Computation. 3, 63–74 (1999)
16. Carter, M.W.: A survey of practical applications of examination timetabling. Operations Research 34, 193–202 (1986)
17. Merlot, L.T.G., Boland, N., Hughes, B.D., Stuckey, P.J.: A hybrid algorithm for the examination timetabling problem. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 207–231. Springer, Heidelberg (2003)
18. Asmuni, H., Burke, E.K., Garibaldi, J.M., McCollum, B.: Fuzzy multiple ordering criteria for examination timetabling. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 334–353. Springer, Heidelberg (2005)
19. Ayob, M., Burke, E.K., Kendall, G.: An iterative re-start variable neighbourhood search for the examination timetabling problem. In: Burke, E.K., Ross, P. (eds.) Practice and Theory of Automated Timetabling. LNCS, vol. 1153, pp. 336–344. Springer, Heidelberg (1996)
20. Carter, M.W., Laporte, G., Lee, S.Y.: Examination timetabling: Algorithmic strategies and applications. Journal of Operational Research Society 47(3), 373–383 (1996)