Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at University Technology MARA

Graham Kendall and Naimah Mohd Hussin

Automated Scheduling, Optimisation and Planning (ASAP) Research Group
School of Computer Science and Information Technology, University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
Email: {gxk, nbh}@cs.nott.ac.uk

Abstract: In this paper we introduce, and solve, an examination timetabling problem at University Technology MARA (UiTM). UiTM is the largest university in Malaysia. It has 13 branch campuses and offers 144 programs delivered by 18 faculties. We discuss their examination timetabling problem with respect to its size, complexity and constraints. We analyse and process their real world data, and produce solutions utilising a tabu search based hyper-heuristic framework. Since this is a new dataset and no solutions have been published in literature, we can only compare our results with an existing manual solution. We find that our solution is at least 80% better with respect to proximity cost. We also compare our approach against a benchmark dataset and show that our method can produce good quality results.

1. Introduction

Work on timetabling problems has been published since the 1960's (Appleby et al 1960), (Csima and Gotlieb 1964). Since then, numerous researchers have been working on problems ranging from sports timetabling (Trick 2000), railway timetabling (Isaai and Singh 2001), (Caprara et al 2001), school timetabling (Abramson et al 1999), (Colorni et a 1998), (Ribeiro and Lorena 2001), (Hansen 1995), (Schaerf and Schaerf 1996) and university timetabling (Awad and Chinneck 1998), (Burke and Newall 1999), (Ergul 1995), (Marti et al 2000), (Ross et al 1998), (Thomson and Dowsland 1998), (Terashima Marin et al 1999), (White & Xie 2000).

Most academic institutions face the problem of scheduling both courses and examinations in every semester or term. As the difficulty of the problem increases, due to a large number of students, courses, exams, rooms and invigilator constraints, an automated timetabling system that can produce feasible and high quality timetables is often required. The timetabling procedure at universities and schools varies from manual timetabling, semi-automated timetabling to fully automated timetabling. A survey conducted by Burke et al (1996) on 56 British universities discovered that only 58% (32 universities) of their respondents use a computer at some stage in producing

their examination timetable and 21% (11 universities) of these have a scheduling system. Only two universities use commercial software whilst the other systems were developed in house. Thus, while commercialised software is available, such as, EXAMINE (Carter 1997 and Paechter et al 2000), SyllabusPlus (Forster 1995), ConBaTT (Goltz and Matzke 2000), OPTIME (McCollum and Newall 2000), and CelCAT (Rogalla 1997), many universities are yet to be convinced that an automated system will provide a satisfactory solution. Universities may need to develop their own system or to customise a commercial system to fulfill their specific needs and requirements in timetabling. Once a customised system is developed, it will also require full support with frequent update and maintenance due to changes in academic policy or education structure.

An early survey by Comm and Mathaisel (1988) involving 1494 U.S. college registrars concluded that there was a large market for a computerised timetabling system and most registrars were unhappy with their current systems. The survey conducted by Burke et al (1996) received feedback from 56 registrars of British Universities with regards to the nature of their examination timetabling problem, how they solved it (manual or automated) and what qualities were considered for a good examination timetable. The survey showed that a computerised system must produce good quality timetables allowing some user intervention, it must be easy to use and comprehensive and compatible with any previous systems. JISC (Joint Information System Committee) Technology Applications Programme (JTAP 1998) published findings from a questionnaire from which they received replies from 16 universities in the U.K. The universities were asked whether a central computerised system was in use and for their views on its effects. The report concluded that centralising the whole process of room bookings, examination timetables and lecture timetabling was carried out in phases using a wide variety of software packages and there was a need for a full and complete management support for such systems.

The above surveys show that a lot of computerisation has taken place over the years and there is ongoing research on new techniques and methods to solve timetabling problems so as to produce better quality solutions. In this paper, we will focus on examination timetabling problem faced by universities. Carter and Laporte (1997) defined the basic problem in examination timetabling as "*the assigning of examinations to a limited number of available time periods in such a way that there are no conflicts or clashes*". A timetable is feasible if all examinations can be scheduled and all other hard constraints are not violated. Student conflict exists if at least one student is scheduled to sit for more than one exam at the same time. This conflict is categorised as a hard constraint and should be eliminated in an examination timetable. A proximity constraint is an example of a soft constraint that can be violated if necessary. A weighted proximity cost is assigned whenever a student has to sit for two examinations scheduled at most 5 slots apart. A lower proximity cost means that we have a good quality solution and our objective is to minimise the proximity cost. Carter, Laporte and Lee (1996) have a set of benchmark examination timetabling instances from real examination problem datasets from various universities all over the world. They applied a variety of constructive algorithms, with backtracking, based on graph colouring heuristics and they solve the problem

with and without capacity constraint (capacitated and un-capacitated problem). The solution quality is measured by an objective function based on proximity cost. A lower proximity cost indicates that, on average, a student will have his/her examinations spread over the length of his/her examination period and, therefore, will have more time to concentrate on each exam. Burke, Newall and Weare (1996) introduced another dataset from the University of Nottingham that also includes room requirements and capacities. They used a memetic algorithm and a weighted objective function for adjacent, overnight and unscheduled exams. Burke, Newall and Weare (1998) apply some sequential heuristics and various ordering techniques to allocate exams to slots while not violating the clash and capacity constraints. Merlot et al (2002) introduced two new datasets from the University of Melbourne that also includes two additional hard constraints: exam availability (exams preassigned to specific slots) and large exams (large exams scheduled in the first *n* slots). They used a hybrid algorithm (constraint method, simulated annealing and hill climbing) to find a feasible schedule and found that they have to relax the constraints (by adding additional slots to the large exams constraints) so as to produce a clash free timetable.

All the datasets above are made public via ftp://ftp.mie.utoronto.ca/pub/carter/testprob and ftp://ftp.cs.nott.ac.uk/ttp/Data and http://www.or.ms.unimelb.edu.au/timetabling.

We presented the examination timetabling problem from the University Technology MARA in Cowling, Kendall and Mohd Hussin (2002). University Technology MARA (UiTM) is the largest university in Malaysia with a total number of students approaching 100,000. The university has 13 branch campuses, one in every state in Malaysia with 144 programs offered by 18 faculties. A common examination timetable is shared amongst all campuses and programs since students sitting for the same examination paper must take it at the same time, irrespective of their geographical location.

Wan Ya and Baharudin (2001) from UiTM uses an examination scheduling program (developed in house, using the COBOL language, about 30 years ago) to produce a first draft of the examination timetable. The timetable then goes through a manual update process by scheduling new courses and removing old ones.

Apart from the constraints that are common for examination timetabling (Carter 1986; Carter and Laporte 1997; Schaerf 1999), UiTM has to consider the following additional constraint: If an exam falls on a state public holiday and there are students from that state sitting for the exam then the exam must be moved to another slot. Malaysia has a number of public holidays that are not shared between states.

Data on available space for exams is not considered when moving exams and therefore every faculty, centre and branch campus needs to give prompt feedback on space availability for exams already scheduled. Once the examination timetable is in its final draft, it is sent to all faculties and branch campuses for the assignment of rooms and invigilators.

In the next section, we present a detailed description of the examination timetabling problem at University Technology MARA. Section 3 provides a description of our tabu search based hyper-heuristic and Section 4 shows how our algorithm works with this new large dataset and how it compares with the only other known solution to this problem. Section 5 concludes with a summary and presents future research directions.

2. UiTM Examination Timetable Problem Formulation

2.1 Problem Description

University Technology MARA (UiTM) is the largest university in Malaysia with 13 branch campuses. The Center for Integrated Information System (CIIS), UiTM has the responsibility for planning and managing the overall Information Technology strategy in UiTM and fulfilling administrative and academic needs. One of its information system modules, Integrated Student Information Systems (iSIS), was designed and developed as a collaborative project. This system offers 6 main modules encompassing the complete Student Life Cycle process, from Intake to Convocation and Alumni. The 4 principal modules comprise the Student Intake System, Academic Affairs Systems, Student Affairs System and Student Accounting System. Thirteen personnel headed by a senior information system officer provide support for the system and two information system officers are specifically assigned to the examination unit.

No.	Branch Campuses	No. of Students	No. of Student Exams	Avg Exam Per Student
1.	Shah Alam	40,275	222,559	5.52
2.	Melaka	4,447	25,334	5.70
3	Negeri Sembilan	315	1,913	6.07
4.	Johor	4,057	22,109	5.45
5.	Perak	5,366	31,518	5.87
6.	Perlis	5,824	32,807	5.63
7.	Kelantan	3,515	19,627	5.58
8.	Terengganu	4,842	27,444	5.67
9.	Pahang	4,607	27,221	5.91
10.	Sarawak	4,800	25,618	5.34
11.	Sabah	2,423	11,470	4.73
12.	Penang	1,453	9,296	6.40
13.	Kedah	2,751	15,285	5.56
	UiTM-03 (Total)	84,675	472,201	5.56

Table 1: Characteristics of UiTM Dataset

We have been fortunate to get most of our examination data from the Center for Integrated Information System (CIIS). The data is classified by faculty and branch campuses. The total number of students enrolled in Semester 2, 2002/2003 was 84,675, course enrolment was 472,201 and total number of courses to be scheduled was 2,650. (see Table 1). The data supplied from CIIS was in the form of, a list of exams that must be scheduled, a list of exams that must be scheduled at the same time (concurrent), a list of students and their course selection (split by campuses) and an examination timetable that was used in the May 2003 semester. We do not have any information regarding capacity or other hard constraints that were imposed via feedback by faculties or campuses. The original list of exams has 2,650 exams to be scheduled, but after computing the enrolment for each examination using the student file, we had to remove examinations with zero enrolments even though these examinations were present in the examination timetable. Examinations with zero enrolments will have no effect in the way we compute solution quality and therefore can be removed from the timetable. We removed 491 exams and we only need to schedule 2,159 examinations. It is understandable that the supplied student list might not be accurate and up to date because the data processing involves many faculties and campuses.

2.2 Problem Formulation

We can represent the examination timetabling problem as follows:

- E: A set of *m* examinations E_1, E_2,…..E_m
- S: A set of *n* slots S_1, S_2,…..S_n
- U: A set of u campus U_1, U_2,…..U_u
- A final exam timetable T_{mn} such that: T_{ik} = 1 if exam *i* is scheduled in slot *k*, 0 otherwise.
- CampusType = {A, B} where campus type A has half-day Saturday and full-day Sunday weekend and campus type B has half-day Thursday and full-day Friday weekend.
- A conflict matrix C_{mm} such that: C_{ij} = total number of students sitting for both exams *i* and *j* categorised by campus type.
- A co-schedule matrix R_{mm} such that: R_{ik} = 1 if exam *i* and exam *k* must be scheduled in the same time slot, 0 otherwise.

The examination timetabling problem is to assign examinations to *n* number of slots subject to various constraints, so as to minimise various costs. The total number of slots is already fixed and examination scheduler must schedule examinations into at most *n* slots. There are 2,159 examinations with some exams being held concurrently. Even though each exam may have a different duration (120, 150 or 180 minutes), we will treat each exam as occupying a complete slot of 180 minutes. The total number of exam days is 20 and each day will have 2 slots (morning and afternoon), i.e. we

will have 40 slots in which to assign examination. We categorise each campus (A or B) to indicate which days or slots must not be used in assigning exams taken by students in that particular campus. Campuses in category A has weekend: half-day on Saturday and full-day on Sunday while campuses in category B has weekend: half-day on Thursday and full-day on Friday. The exam dates were from 20th April 2003 to 10th May 2003 with 1st May as a public holiday across all campuses, so no exams can be scheduled on this day. For this dataset, we are not concerned with other public holidays since none occur on a different day for different states. On other occasions there might be a situation where a campus has a public holiday and others do not.

2.2.1 Constraints

Hard constraints are those which cannot be violated. A timetable that violates a hard constraint will render it infeasible. Such infeasibilities may be unavoidable in certain cases and universities have to take drastic measures to resolve the problem.

The following hard constraints are present in the UiTM dataset:-

a. Conflict: No student should sit for more than one exam in the same slot.

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{n} T_{ik}T_{jk}C_{ij} = 0$$

If exam *i* and exam *j* are scheduled in slot *k*, the number of students sitting for both exam *i* and *j* (C_{ij}) must be equal to zero, and this should be true for all exams already allocated.

b. Co-schedule: Exams that must be scheduled together must be assigned the same time slot.

For all exams *i*:

$$\sum_{j=1}^{m}\sum_{k=1}^{n} T_{ik}T_{jk}R_{ij} = \sum_{j=1}^{m} R_{ij}$$

All exams that must be scheduled with exam *i* should be assigned to the same slot *k*.

c. All exams must be scheduled:

$$\sum_{k=1}^{n} T_{ik} = 1 \qquad\qquad i = 1..\text{m}$$

Each exam, (E_1, E_2,…..E_m) should be scheduled only once.

2.2.2 Costs

Other additional soft constraints that are specific to university requirements can be added to this problem. We determine the cost of an examination timetable solution based on the penalty given if certain soft constraints are violated. The soft constraints that we consider are as follows:

a. **Proximity cost:** A proximity cost x_s is given whenever a student has to sit for two examinations scheduled s periods apart: these weights are $x_1 = 16$, $x_2 = 8$, $x_3 = 4$, $x_4 = 2$ and $x_5 = 1$. P_{ik} is the proximity cost if exam i is scheduled in slot k.

The total proximity cost of a timetable is as follows:

$$\sum_{i=1}^{m} \sum_{k=1}^{n} T_{ik} P_{ik}$$

b. **Weekend cost**: The current examination timetable scheduler schedules exams during the weekend. We can try to produce a better quality timetable by penalising exams that are scheduled during the weekend. Weekends for category A campuses is half-day on Saturday and full-day on Sunday and weekends for category B campuses is half-day on Thursday and full-day on Friday. A penalty cost of 16 is given whenever a student has to sit for a weekend exam. W_{ik} is the proximity cost if exam i is scheduled in slot k.

The total weekend cost is as follows:

$$\sum_{i=1}^{m} \sum_{k=1}^{n} T_{ik} W_{ik}$$

Finally, our objective is to optimise the following:

$$\sum_{i=1}^{m} \sum_{k=1}^{n} T_{ik} P_{ik} \qquad + \sum_{i=1}^{m} \sum_{k=1}^{n} T_{ik} W_{ik}$$

i.e. minimise the total proximity cost and the weekend cost.

3. Tabu Search Based Hyper-heuristic

A hyper-heuristic (Burke et al 2003) works at a higher level of abstraction than a meta-heuristic and does not require domain knowledge. A hyper-heuristic only has access to non-domain specific information that it receives from the heuristics that it operates upon. A hyper-heuristic can be implemented as a generic module that has a common interface to the various low-level heuristics and other domain specific knowledge (typically the evaluation function) of the problem being solved. Initially, the hyper-heuristic needs to know the number of *n* heuristics provided by the low-level heuristic module. It will guide the search for good quality solutions by setting up its own strategy for calling and evaluating the performance of each heuristic known by their generic names H_1, H_2, …. H_n. The hyper-heuristic does not need to know the name, purpose or implementation detail of each low-level heuristic. It just needs to call a specific heuristic, H_i, and the heuristic may modify the solution state and return the result via an evaluation function. The low-level-heuristic module can be viewed as a 'black box' that hides the implementation detail and only returns a new solution and a revised evaluation function.

3.1 Hyper-heuristic module

The hyper-heuristic module is the focus of this research, where we need to design and test strategies that can intelligently select the best heuristic that will help guide the search to either intensify or diversify the exploration of the search region.

The general framework for our hyper-heuristic is as follows:

```
Step 1.   Construct initial solution
Step 2.   Do
                Consider heuristics that are not tabu
                Apply chosen heuristic and make the heuristic tabu
                Update solution
          Until terminating condition
```

The initial solution is produced using a constructive heuristic (largest degree or saturation degree (Carter and Laporte (1997))). Next, a randomisation (randomly move exams to other valid slots) is carried out in order to start different runs with different solutions. In Step 2 we explore the neighbourhood to search for a better solution or local optima (and possibly global optima). The framework is similar to a local search except that in Step 2, we explore the neighbourhood by selecting which heuristic to use to update the current solution.

The core part of the algorithm is Step 2, where we need to decide which heuristics should be considered and which heuristic should be applied. Each heuristic differs in

how it decides to move, thus creating its own search space region (heuristic search space) in the solution search space. In the search for good quality solutions, the hyper-heuristic exhibits a kind of reinforcement learning that will assist in an intelligent action at each decision point. At this point, the hyper-heuristic can actually choose intelligently when to intensify or diversify the search because we believe that by allowing the low-level heuristics to compete at each iteration and selecting the heuristic with the best performance will help to balance the diversification and intensification of the solution search space. Heuristics that have been applied become tabu so that in the next iteration we can explore the solution space of other low-level heuristics that may perform well but, perhaps, not as well as the previous heuristics that are now tabu.

The hyper-heuristic monitors the behaviour of each low–level heuristic by storing information about their performance using an adaptive memory. Our hyper-heuristic uses a tabu list that is of a fixed length n, where n is the number of low-level heuristics. Instead of storing moves, each tabu entry stores (non-domain) information about each heuristic i.e. heuristic number, recent change in evaluation function, CPU time taken to run the heuristic, and tabu status (or tabu duration, which is the term we use here). Tabu duration indicates how long a heuristic should remain tabu and, will therefore, not be applied in the current iteration. If the tabu duration is zero, the heuristic is said to be tabu inactive and can be applied to update the solution. If the tabu duration is non-zero, the heuristic is said to be tabu active and may not be used to update the solution. The tabu duration is set for a heuristic whenever a tabu restriction is satisfied. After each iteration, the tabu duration is decremented until it reaches zero and the heuristic is now tabu inactive. We do not use any aspiration criteria since a tabu active heuristic will have its tabu duration decremented in each iteration, and will eventually be tabu inactive. If all heuristics are tabu active in any iteration, no heuristics will be evaluated and obviously none will be applied. Therefore, a heuristic changes its status from tabu active to tabu inactive only when the tabu duration is zero. In using a tabu list, we need to decide what tabu duration value works best for a given problem instance. Our first implementation used a deterministic tabu duration where, in each run, we used a fixed range of tabu duration and compare the final best solution. In our previous paper, (Kendall and Mohd Hussin 2004) we showed empirically, using deterministic tabu duration, that, an effective tabu duration is dependent upon the conflict matrix density of a given examination timetabling problem instance. In general having a low tabu duration allows the exploration to move within the same heuristic search space and a high tabu duration allows exploration into other possible regions. If a tabu duration is too high, we found that the quality of the solution deteriorates since good heuristics are made tabu for too long. If a tabu duration is too low (or equal to zero), we have limited ourselves to search within a small region in the solution space. In this paper, we will consider both deterministic tabu duration and random dynamic tabu duration. Random dynamic tabu duration uses a tabu duration range (t_{min} and t_{max}), and at each decision point of making a heuristic tabu, a tabu duration value is selected at random from the given range.

The next issue we have to address is the mechanism for updating a solution. In each iteration, we compare a solution from each heuristic and take the best solution. We use three different strategies for accepting the best solution as a new solution at each decision point. First, is to accept the solution from the best performing heuristic. This best solution may not improve the current or previous best solution. Second, is to accept the best solution and if this solution improves the previous solution, the same heuristic will be applied until it cannot improve the solution anymore (steepest ascent hill climbing). Third, is to accept the best solution only if the solution is less than a boundary penalty. The boundary penalty was first introduced by Dueck (1993) in his great deluge algorithm, and Burke et al (2004) also apply it to an examination timetabling problem.

We have implemented and tested three different hyper-heuristics, which incorporate the various strategies mentioned above.

- The simplest form, i.e., Simple Tabu Search Hyper-heuristic (TSHH-S), where we consider all tabu inactive heuristics and apply the heuristic that has the best improvement only. The algorithm iterates for a fixed time or until there is no further improvement for a given number of heuristic calls.
- The second hyper-heuristic is Tabu Search Hyper-heuristic with Hill Climbing (TSHH-HC), which adds to TSHH-S a successive call to the best performing heuristic until no further improvement is made.
- The third hyper-heuristic is Tabu Search Hyper-heuristic with Great Deluge (TSHH-GD), which updates a solution within a certain boundary only.

For each of these hyper-heuristics, we apply both deterministic tabu duration and random dynamic tabu duration.

3.2 Low-level heuristics module

Low-level heuristics are heuristics that allow movement through a solution space and that require domain knowledge and are problem dependent. Each heuristic creates its own heuristic search space that is part of the solution search space. The idea is to build a collection of (possibly) simple moves or choices since we would like to provide a library of heuristics that can be selected intelligently by a hyper-heuristic tool.

The heuristics change the current state of a problem into a new state by accepting a current solution and returning a new solution. Each low-level heuristic can be considered as improvement heuristics that returns a move, a change in the penalty function and the amount of time taken to execute the heuristic. The best performing heuristic should cause a maximum decrease in penalty (the lowest value). Each move from an individual heuristic may cause the search to probe into the current neighbourhood or to explore a different neighbourhood. A change in the penalty value means changing the penalty value for each of the soft constraints that were violated (first order conflict, second order conflict, etc) or moving an exam into an unscheduled list (exam becomes unscheduled and violates hard constraints).

We used the same low-level heuristics as in Kendall and Mohd Hussin (2004) i.e.:

- Five graph colouring heuristics that select an exam from an unscheduled list and schedule it into the best available slot that maximises the reduction in penalty. The heuristics are: largest enrolment, largest exam conflict, largest total student conflict, largest exam conflict already scheduled, and exam with least valid slots.
- Five move heuristics that select an exam, either at random, with maximum penalty, with highest second order conflict or highest first order conflict. This exam is rescheduled into a new random slot or a new slot which maximises the reduction in either the total penalty, total first order conflict or second order conflict.
- Two swap heuristics that select an exam, either at random, with maximum penalty or with minimum penalty. The two exams selected will swap slots subject to no hard constraint violations.
- A heuristic that removes a randomly selected exam from examinations already scheduled. This is the only heuristic which will move the search into an infeasible region because any exam may be unscheduled. We make sure that the search can move back into its feasible region by un-scheduling exams that have other valid slots to move to in the next iteration.

All of the above low-level heuristics are either 1-opt (one move) or 2-opt (two moves) and there is also a mixture of some randomness and some deterministic selection of exams and slots. We purposely use low-level heuristics that are simple moves rather than low-level heuristics with intelligence and complex moves because we want to make sure that the hyper-heuristic can recognise good moves and make an intelligent decision based on these simple moves. Furthermore, we want to make the problem-domain knowledge heuristics easy to implement and the hyper-heuristic more generalised.

4. Experimental Results

We have implemented and tested our tabu search based hyper-heuristic framework on a PC with an AMD Athlon 1 GHz processor, 128 Mb RAM and Windows 2000. The program was coded in C++ using an object-oriented approach. We defined and implemented the hyper-heuristic and heuristics as objects that have a common interface and can interact with each other. In our previous paper (Kendall and Mohd Hussin, 2004), we tested the simplest form of our hyper-heuristic module TSHH-B with deterministic tabu duration on Carter's examination timetabling benchmark data. We compared the results (using proximity cost per student) and found that the method is able to find good solution on all datasets. Our tabu search based hyper-heuristic method has also added a significant improvement to tabu search because our results are better in all cases compared to tabu search approach by Di Gaspero and Schaerf (2000). Without changing the low-level heuristics, we tested two more hyper-heuristics with two different objective functions and produced results with respect to minimising proximity cost and minimising both proximity and weekend cost

4.1 Proximity Cost Evaluation Method

We use our previous method and proximity cost evaluation function and run it on our new dataset. We also run our new and improved method: Tabu Search Hyper-heuristic with Hill Climbing (TSHH-HC) and Tabu Search Hyper-heuristic with Great Deluge (TSHH-GD) with Carter's examination timetabling benchmark data and UiTM-03 dataset. Here, we will discuss in how the algorithm performs when applied to UiTM-03 dataset and compare its behaviour with one other dataset i.e car-s-91 (one of the benchmark dataset). Car-s-91 (Carleton University, Ottawa) is one of the largest dataset with 682 exams to be scheduled in 35 slots, 16,925 students and 56,877 student exams.

	UiTM-03			**Car-s-91**		
Tabu Duration	**TSHH-B**	**TSHH-HC**	**TSHH-GD**	**TSHH-B**	**TSHH-HC**	**TSHH-GD**
0	2.16	2.08	2.14	6.88	6.78	6.85
1	1.55	1.55	1.44	6.83	6.88	7.01
2	1.93	1.94	1.37	5.37	5.14	5.15
3	3.95	3.84	**1.35**	6.31	6.04	**4.93**
4	6.37	6.33	1.44	6.91	7.10	5.01
Random	1.65	1.63	1.4	5.43	5.31	5.6

Table 2: Comparison of results between UiTM-03 and Car-s-91 datasets

	TSHH-B	**TSHH-HC**	**TSHH-GD**	**Manual**	**GD**	**SA**
UiTM-03	1.55	1.55	1.35	12.83	1.4	1.68

Table 3: Compare results for UiTM dataset with other solution methods

Referring to Table 2, the best published result for car-s-91 is 4.65 (Burke and Newall 2003) using a combination of adaptive initialisation strategies and great deluge. Our best result for car-s-91 is 4.93 using tabu search based hyper-heuristic and great deluge with tabu duration equal to 3. The first column shows the tabu duration i.e how long a heuristic is placed in the tabu list. We tested with both deterministic tabu duration and random dynamic tabu duration. Our method shows similar behaviour between both datasets, where, the best result is obtained using tabu search hyper-heuristic and great deluge with tabu duration equals 3. Table 3 compares our best result on the UiTM dataset with the existing manual solution, the great deluge algorithm and simulated annealing.

Figure 1 and Figure 2 show the best proximity cost, with various tabu durations obtained for uitm-03 and car-s-91 datasets. In our tabu based hyper-heuristic strategy, we apply the concept of heuristics cooperating with each other rather than penalising a non-performing heuristic. When the tabu duration is greater than zero, we apply a tabu restriction where a heuristic will be tabu active if its solution value has been

accepted to update the current solution. The heuristic will remain tabu active for a number of steps equal to tabu duration. We make a heuristic tabu because we want to direct the search to other possible heuristic search spaces. Eventually we may go back to a heuristic search space once it is no longer tabu active and can give the best solution amongst all tabu inactive heuristics. Similar to the tabu search meta-heuristic, we need to decide which tabu duration (or list size in tabu search) works best for a problem instance. For UiTM dataset (Figure 1), as tabu duration increases, solution quality improves, and once it reaches its best tabu duration, solution quality begins to deteriorate as we increase the tabu duration further. Car-s-91 (Figure 2) shows only a slight difference when tabu duration equals zero and tabu duration equals one for both hyper-heuristics that either incorporate hill climbing or great deluge. In that instance, solution quality is better when tabu duration is zero compared to when tabu duration is one and improves again after that.

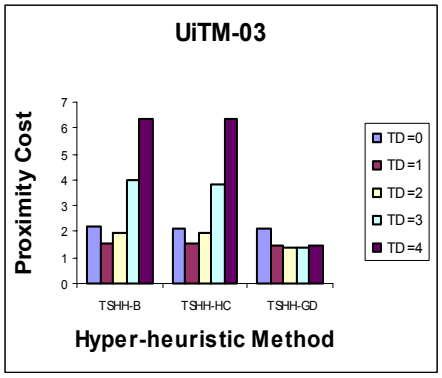

Figure 1: UiTM-03

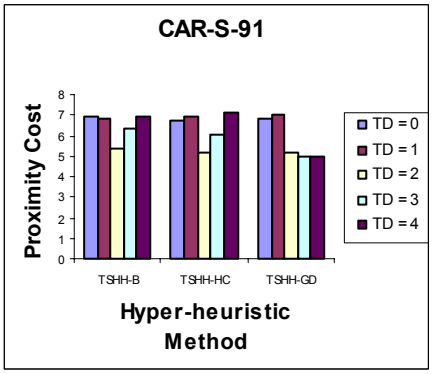

Figure 2: Car-s-91

Figures 3 – 5 show the hyper-heuristic performance with different tabu durations. The graphs show how the hyper-heuristic explores the search space. Both TSHH-S and TSHH-HC show more dispersed points in the graph compared to TSHH-GD because the boundary penalty used in great deluge directs the search to a much better solution. The lower tabu duration does not show much movement in the search space and it might be trapped in local optima, and as we increase the tabu duration, it shows more movement and exploration of search space and therefore able to find a better solution. As the tabu duration increases it becomes more difficult to get better solutions since many heuristics stay tabu too long and the hyper-heuristic has no option but to take the best solution from the worst heuristics. TSHH-GD is better at not moving to a worse solution since it is also directed by the boundary penalty and thus will not make bad moves in such a way that it cannot get back to good solution space. Figure 6 shows a comparison of the TSHH-GD with Great Deluge Algorithm. We ran both algorithms for 4 hours and tracd the penalty evaluation at every 5,000 iterations (steps). The TSHH-GD converges faster in terms of steps but it actually takes longer because each iteration explore various heuristics solution space and may take 10 times longer compared to the Great Deluge Algorithm.

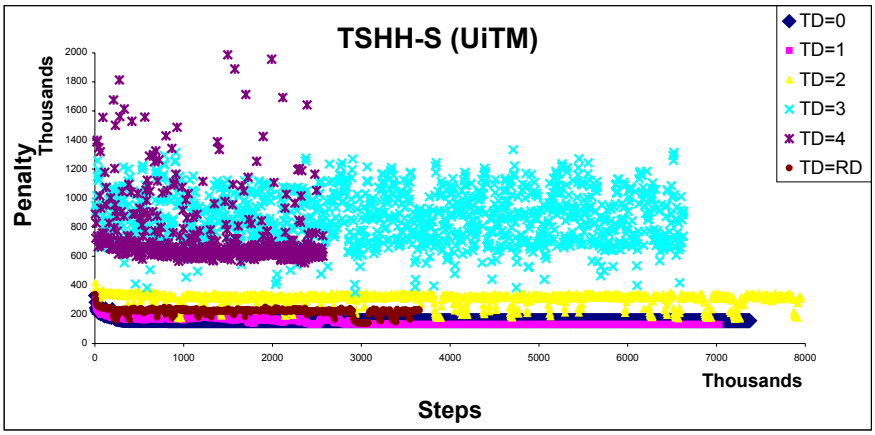

Figure 3: Simple Tabu Search Based Hyper-heuristic for UiTM dataset

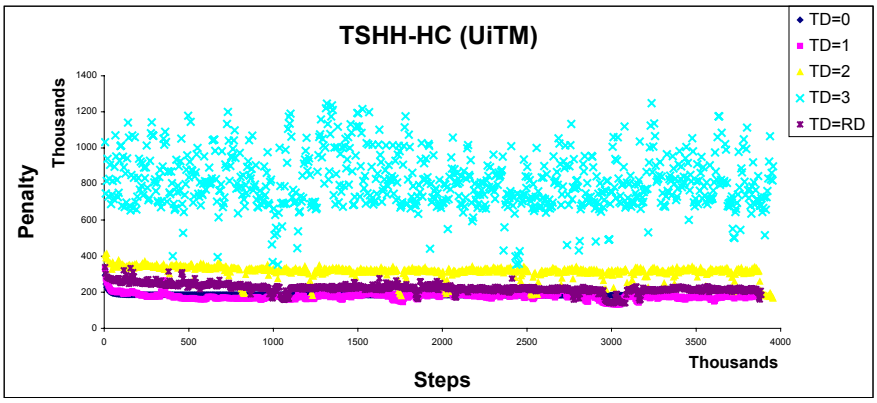

Figure 4: Tabu Search Based Hyper-heuristic with Hill Climbing for UiTM dataset

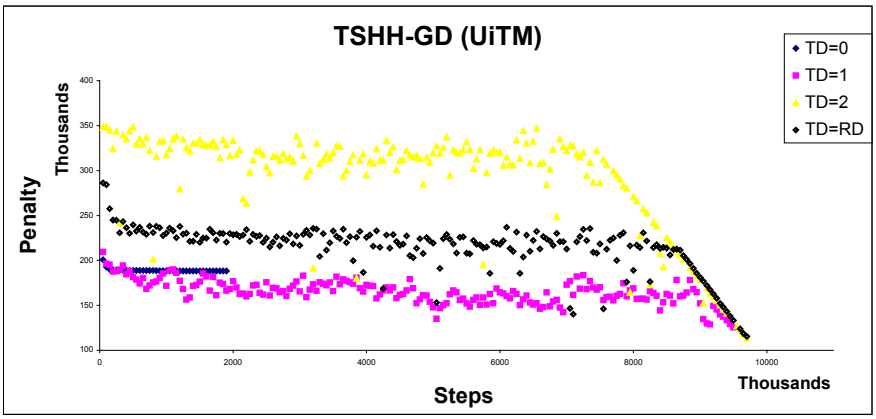

Figure 5: Tabu Search Based Hyper-heuristic with Great Deluge for UiTM dataset

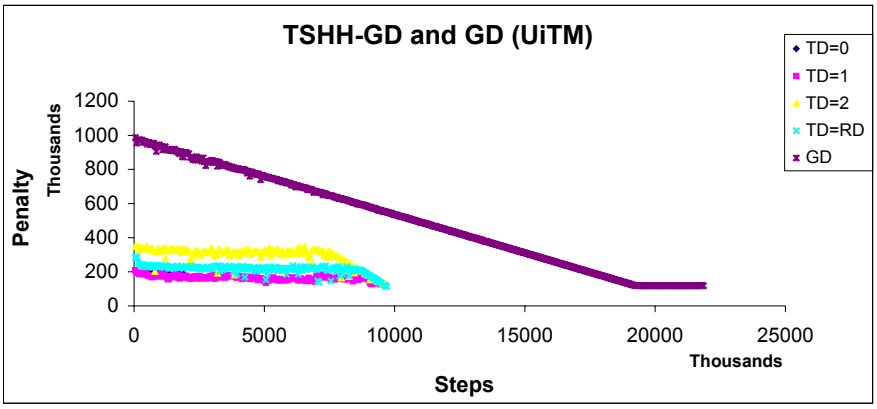

Figure 6: Comparing TSHH-GD and Great Deluge Algorithm performance

4.2 Proximity and Weekend Cost Evaluation Method

The UiTM dataset has an additional constraint, i.e., weekend cost that we want to minimise. We combine the proximity cost and weekend cost in one objective function with the same weights for proximity cost and a weight of 16 for every student who has to sit for a weekend exam. We do not need to do much modification to our hyper-heuristic except for a change in the evaluation function, with weights adjusted for weekend and unscheduled exam (new weight of 10,000). In the previous implementation we refer to second order conflict as a conflict for a student having to sit for exams, 2 slots apart. In this implementation we refer to second order conflict as a conflict for a student who has to sit for a weekend exam. In general, we can further improve our low-level heuristics by generalising the number of conflicts and the type of conflicts we want to minimise. In this way, we can use our general method to solve an examination timetabling problem with multiple objectives. Petrovic and Bykov (2002) solve examination timetabling with multiple objectives by dynamically changing the weights of criterion during the search process. Our method does this implicitly because the hyper-heuristic can choose between the low-level heuristic that minimise the first order conflict or the second order conflict. The only parameter that we need to adjust would be the most suitable weights associated with the conflicts. In the UiTM problem, we can also determine how important it is to minimise first order conflicts or second order conflicts by associating appropriate weights. In our experiments, we tried using 32 and 16 as weekend weights and 5,000 and 10,000 as unscheduled weights. We report here the results for 16 as weekend weights and 10,000 as unscheduled weight. A 5,000 unscheduled weight is not suitable because the total number of students is too large and this will direct the search into an infeasible region since the *remove exam* will outperform other heuristics by un-scheduling an exam.

	UiTM-03		
Tabu Duration	**TSHH-B**	**TSHH-HC**	**TSHH-GD**
0	10.10	9.29	12.12
1	9.27	11.20	11.46
2	9.87	9.70	10.04
3	17.36	17.31	**7.12**
4	16.42	17.16	7.92
Random	9.51	9.65	8.04

Table 4: UiTM-03 that minimised proximity and weekend cost

Table 4 shows the same hyper-heuristic being applied but with a different objective function. The results should be higher than in Table 2 since we are also taking into account the penalty for weekend exams. On average we could say that students may not have to sit for weekend exams or sit for an adjacent exam since our best solution (7.12) is less than the weight for a weekend exam or adjacent exams. But, this is not true since we do need to look at the overall timetable and count the number of students who have to sit for adjacent exams or weekend exams. Our general method is also able to find feasible solutions to the timetabling problem even though in the search process it does un-schedule exams and re-schedule them back again. This is achievable implicitly with the help of our low-level heuristics that specialise in scheduling unscheduled exams.

5. Conclusion

Collecting data from a large university is a difficult task, but with the help of a distributed network and a frequent update of student information, we can facilitate the automated process of producing examination schedule that is feasible and may satisfy everybody involved. Our tabu search based hyper-heuristic has been shown to produce feasible and good quality solutions on other benchmark dataset (Kendall and Mohd Hussin 2004). We compare our results from UiTM dataset with existing manual solution and find that our solution is at least 80% better with respect to proximity cost.

Currently, we are experimenting on an adaptive tabu strategies of selecting best tabu duration at each decision point, and applying other methods on the UiTM-03 dataset.

Acknowledgements

This research is done at University of Nottingham and supported by the Malaysian Public Services Department and University Technology MARA (UiTM), Malaysia

References

1 Abramson D.A., Dang H., and Krisnamoorthy M (1999), "Simulated Annealing Cooling Schedules for the School Timetabling Problem", Asia-Pacific Journal of Operational Research, V 16, pp 1-22.

2 Appleby J.S., Blake D.V. and Newman E.A. (1960), "Techniques for Producing School Timetables on a Computer and their Application to other Scheduling Problems", The Computer Journal V 3, pp 237-245.

3 Awad R., Chinneck J. (1998), "Proctor Assignment at Carleton University", Interfaces vol 28 no 2, pp 58-71.

4 Burke E.K. and Newall J.P. (2003), "Enhancing Timetable Solutions with Local Search Methods", The Practice and Theory of Automated Timetabling (eds Burke E.K. and De Causmaecker P), Lecture Notes in Computer Science Vol. 2740, Springer, pp 195-206.

5 Burke E.K. and Newall J.P. (1999), "A Multi-Stage Evolutionary Algorithm for the Timetable Problem", the IEEE Transactions on Evolutionary Computation, Vol 3.1, pp. 63-74.

6 Burke E.K., Hart E., Kendall G., Newall J., Ross P and Schulenburg S. (2003), "Hyper-Heuristics: An Emerging Direction in Modern Search Technology". Ch 16; Handbook of Meta-Heuristics (eds. Glover F. and Kochenberger), pp 457-474, Kluwer.

7 Burke E.K., Elliman D.G., Ford P.H. and Weare R.F. (1996), "Examination Timetabling in British Universities - A Survey",The Practice and Theory of Automated Timetabling (eds EK Burke and P Ross), Lecture Notes in Computer Science Vol. 1153, Springer,pp 76-92.

8 Burke E.K., Newall J.P., Weare R.F. (1998), "Initialization Strategies and Diversity in Evolutionary Timetabling", IEEE Transactions on Evolutionary computation, Vol 6.1, pp 81 - 103,by the Massachusetts Institute of Technology.

9 Burke E.K., Newall J.P., Weare R.F. (1996), "A Memetic Algorithm for University Exam Timetabling", Proceedings of the 1st Intl. Conference on the Practice and Theory of Automated Timetabling PATAT 1, pp 496-503.

10 Burke, E.K., Bykov, Y., Newall, J., Petrovic, S. (2004), "A Time-Predefined Local Search Approach to Exam Timetabling Problems". Accepted for publication in IIE Transactions on Operations Engineering.

11 Caprara A., Fischetti M., Guida P.L., Monaci M., Sacco G., Toth P., (2001), "Solution of real-world train timetabling problems", Proceedings of the 34th Annual Hawaii International Conference on System Sciences, pp 1057 -1066.

12 Carter M.W. (1986), "A Survey of Practical Applications of Examination Timetabling Algorithms", Operations Research Society of America, Vol 34 No 2 March-April.

13 Carter M. 1997, "EXAMINE:A General Examination Timetabling System", PATAT '97 Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling, pp 363.

14 Carter M.W., Laporte G. and Lee S.Y. (1996), "Examination Timetabling: Algorithmic Strategies and Applications", Journal of the Operational Research Society Vol 47 Issue 3, pp 373-383.

15 Colorni A., Dorigo M. and Maniezzo V. (1998). "Metaheuristic for high-school timetabling" Computational Optimization and Applications, 9, Kluwer Acad. Publ., Dodrecht, NL, pp. 275-298.

16 Comm C.L. and Mathaisel D.F.X (1988), "College Course Scheduling. A Market for Computer Software Support,", Journal of Research of Computing in Education, vol 21, ppp 187-195.

17 Cowling P., Kendall G., Mohd Hussin N. (2002), "A Survey and Case study of Practical Examination Timetabling Problems ", Proceedings of the 4th International

Conference on the Practice and Theory of Automated Timetabling (PATAT2002), Gent, 2002, pp 258-261.

18 Csima J. and Gotlieb C.C. (1964), "Tests on a Computer Method for Construction School Timetables", Communication of the ACM, Volume 7, Number 3, pp 160-163.

19 Dueck G. (1993), "New optimization heuristics for the Great Deluge Algorithm and the Record--to--Record Travel". Journal of Computational Physics, volume 104, pp 86--92.

20 Ergul A. (1995), "GA-Based Examination Scheduling Experience At Middle East Technical University", Proceedings of the 1st Intl. Conference on the Practice and Theory of Automated Timetabling PATAT 1, pp 351-363.

21 Forster G. (1995), "Syllabus Plus: A state-of-the-art planning and scheduling system for Universities and Colleges", Proceedings of the 1st Intl. Conference on the Practice and Theory of Automated Timetabling PATAT 1, pp 244-252.

22 Goltz H.-J., Matzke D. (2000), "ConBaTT - Constraint-Based Timetabling", PATAT 2000 Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, Edmund K Burke and Wilhelm Erben (editors), pp 491.

23 Hansen M.P. (1995), "Planning of High School Examinations in Denmark", European Journal of Operational Research, Vol. 87, pp. 519-534.

24 Isaai M.T., Singh M.G., (2001) , "Hybrid applications of constraint satisfaction and meta-heuristics to railway timetabling: a comparative study", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume: 31 Issue: 1, pp: 87-95

25 JTAP (1998), "Central Timetabling By Computer: A review of existing information" Report by JISC Technology Applications Programme, June 1998 url: http://www.jtap.ac.uk

26. Kendall G., Mohd Hussin N. (2004), "An Investigation of a Tabu Search Based Hyper-heuristic for Examination Timetabling", Accepted for publication in selected papers from MISTA 2003, Kluwer Publication, Kendall G., Burke E. and Petrovic S. (eds). An earlier version of this paper also appeared in the Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA2003) pp 226-233.

27 Marti R., Lourenco H., Laguna M. (2000), "Assigning proctors to exams with scatter search", Computing Tools for Modeling, optimization and Simulation, M. Laguna & J.L. Gonzalez Velarde (Eds), Kluwer Academic publishers, pp 215-227.

28 McCollum B., Newall J. (2000), "Introducing Optime: Examination Timetabling Software", PATAT 2000 Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, Edmund K Burke and Wilhelm Erben (editors), pp 485.

29 Merlot L.T.G., Boland N, Hughes B.D. and Stuckey P.J. (2002), "A Hybrid Algorithm for the Examination Timetabling Problem", The Practice and Theory of Automated Timetabling (eds Burke E.K. and De Causmaecker P), Lecture Notes in Computer Science Vol. 2740, Springer, pp 207-231.

30 Petrovic S. and Bykov Y., (2002), "A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories", The Practice and Theory of Automated Timetabling (eds Burke E.K. and De Causmaecker P), Lecture Notes in Computer Science Vol. 2740, Springer, pp 181-194.

31 Ribeiro Filho G. and Lorena L. A. N (2001). A Constructive Evolutionary Approach to School Timetabling. In Applications of Evolutionary Computing , Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H., (Eds.) - Springer Lecture Notes in Computer Science vol. 2037, pp. 130-139 .

32 Rogalla S. (1997), "CELCAT:A Practical Solution to Scheduling Problems", Corbett Engineering, UK 368 PATAT '97 Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling, pp 368.

33 Ross P.M., Hart E. and Corne D. (1998), ``Some observations about GA-based exam timetabling", Practice and Theory of Automated Timetabling II, eds E.Burke and M.Carter, pp. 115-129, Springer-Verlag Lecture Notes in Computer Science 1408, Springer-Verlag.

34 Schaerf A. and Shaerf M. (1996), "Local Search Techniques for high school timetabling",Proceedings of the 1st Intl. Conference on the Practice and Theory of Automated Timetabling PATAT 1 and The Practice and Theory of Automated Timetabling (eds EK Burke and P Ross), Lecture Notes in Computer Science Vol. 1153, Springer-Verlag.

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

36 Terashima-Marín H., Ross P.M., and Valenzuela-Rendón M. (1999), "Application of the Hardness Theory when Solving the Timetabling Problem with GAs", Proceedings of the Congress on Evolutionary Computation 1999 Washington, D.C., July 6-9,99, pp 604-611

37 Thomson J.M., Dowsland K.A. (1996), "Variants of simulated annealing for the examination timetabling problem", Ann. Oerational Research pp 105-128.

38 Trick M.A. (2000), "A Schedule-then-Break Approach to Sports Timetabling", Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling PATAT 2000, pp. 128-147, Konstanz, 2000The Practice and Theory of Automated Timetabling: Volume Three, E.K. Burke and W.Erben (eds) pp 242-253.

39 Wan Ya and Baharudin N. (2001), Interview with Manager and System Analyst, Examination Unit, Center for Integrated Information System, University Technology MARA, August 2001.

40 White G.M. and .Xie B.S. (2000), "Examination Timetables and Tabu Search with Longer Term Memory", Practice and Theory of Automated Timetabling III (eds Burke E.K. and Erben W.), Lecture Notes in Computer Science, 2079, Springer, pp. 85-103.