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

Graham Kendall[1] and Naimah Mohd Hussin[2]

[1] 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
gxk@cs.nott.ac.uk

[2] Faculty of Information Technology and Quantitative Sciences, MARA University of Technology, 40450 Shah Alam, Selangor Darul Ehsan, Malaysia
naimah@tmsk.uitm.edu.my

**Abstract.** In this paper we introduce an examination timetabling problem from the MARA University of Technology (UiTM). UiTM is the largest university in Malaysia. It has 13 branch campuses and offers 144 programmes, delivered by 18 faculties. This dataset differs from the others reported in the literature due to weekend constraints that have to be observed. We present their examination timetabling problem with respect to its size, complexity and constraints. We analyse their real-world data, and produce solutions utilising a tabu-search-based hyper-heuristic. Since this is a new dataset, and no solutions have been published in the 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 is able to produce good quality results.

## 1 Introduction

Work on timetabling problems has been published since the 1960s [17]. Since then, numerous researchers have been working on problems ranging from sports timetabling [35], railway timetabling [14], [25] and educational timetabling [2], [3], [6], [15], [17], [30].

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, examinations, 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. [10] received feedback from 56 registrars from British universities with regard

E. Burke and M. Trick (Eds.): PATAT 2004, LNCS 3616, pp. 270–293, 2005.

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. They discovered that only 58% (32 universities) of their respondents used a computer at some stage in producing their examination timetable and 21% (11 universities) of these had a scheduling system. Only two universities used commercial software whilst the other systems were developed in house. Thus, while commercialised software is available, such as, EXAMINE [16], SyllabusPlus [23], ConBaTT [24], OPTIME [28], and CelCAT [32], 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 customise a commercial system to fulfil their specific needs. Once a customised system is developed, it will also require full support with frequent updates and maintenance due to changes in academic policy or educational structures.

An early survey by Comm and Mathaisel [19] in 1988 involving 1494 U.S. college registrars concluded that there was a large market for a computerised timetabling system and that most registrars were unhappy with their current systems. The survey showed that a computerised system must produce good quality timetables allowing some user intervention, it must be easy to use and be comprehensive and compatible with any previous systems. JISC (Joint Information System Committee) Technology Applications Programme [26] published findings from a questionnaire from which they received replies from 16 universities in the UK. The universities were asked whether a central computerised system was in use and for their views on its effectiveness. 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 full and complete management support for such systems.

The above surveys show that much 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 focus on the examination timetabling problem faced by universities. Carter and Laporte [17] defined the basic problem 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 examination at the same time. This conflict is categorised as a hard constraint and should be eliminated. 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 five slots apart. A lower proximity cost relates to a better quality solution and thus our objective is to minimise the proximity cost.

Carter et al. [18] provide a set of benchmark examination timetabling instances from real examination problem datasets from universities across the world. They applied a variety of constructive algorithms, with backtracking, based on graph colouring heuristics and solve the problem with and without

capacity constraints (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 better spread over the length of his/her examination period and, therefore, will have more time to concentrate on each examination.

Burke et al. [12] 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 examinations. Burke et al. [11] apply some sequential heuristics and various ordering techniques to allocate examinations to slots whilst not violating the clash and capacity constraints.

Merlot et al. [29] introduced two new datasets from the University of Melbourne that includes two additional hard constraints: examination availability (examinations preassigned to specific slots) and large examinations (large examinations scheduled in the first $n$ slots). They used a hybrid algorithm (a 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 examinations constraints) so as to produce a clash free timetable.

All of the datasets described above are available via

- ftp://ftp.mie.utoronto.ca/pub/carter/testprob
- 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 MARA University of Technology in [20]. MARA University of Technology (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 programmes offered by 18 faculties. A common examination timetable is shared amongst all campuses and programmes since students sitting for the same examination paper must take it at the same time, irrespective of their geographical location.

UiTM uses an examination scheduling program [36] (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 [8], [15], [17], [33], UiTM has to consider the following additional constraint: If an examination falls on a state public holiday and there are students from that state sitting for the examination then the examination must be moved to another slot. Malaysia has a number of public holidays that are not shared between states. Data on the available space for examinations are not considered when moving examinations and therefore every faculty, centre and branch campus needs to provide prompt feedback on space availability for examinations 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 this paper, we apply a tabu-search-based hyper-heuristic approach to the UiTM examination timetabling problem. This method has been used to solve

the Carter benchmark datasets and has produced competitive results compared to other methods [27]. Hyper-heuristic [7], [9] methods operate at a higher level of abstraction than other search methods and are able to intelligently choose a (meta-)heuristic to be applied at any given time. We refer to Burke et al. [7] for further motivation and discussion on the emergence of hyper-heuristics in solving optimisation problems. An example of a hyper-heuristic approach for solving a large-scale university examination timetable can be seen in Terashima-Marin et al. [34]. The problem is solved in phases. Each phase uses a different set of heuristics and a switch condition determines when to move from one phase to the other. A genetic algorithm, using an indirect chromosome representation, was used to evolve the choice of heuristics and the switch condition. Another recent application of hyper-heuristic on timetabling problems is presented Burke et al. [5]. The hyper-heuristic uses tabu search to search for different permutations of graph heuristics that solve both examination and course timetabling problems. A further use of a hyper-heuristic is in solving multi-objective [4] space allocation and timetabling problems. The idea is to choose, at each iteration, a heuristic that is suitable for the optimisation of a given, individual objective.

Tabu search is a meta-heuristic that has been applied successfully on examination timetabling problem [21], [37]. In this paper, we incorporate the tabu list mechanism in our hyper-heuristic framework to help guide the selection of heuristics.

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

## 2 UiTM Examination Timetable Problem Formulation

### 2.1 Problem Description

MARA University of Technology (UiTM) is the largest university in Malaysia with 13 branch campuses. The Centre for Integrated Information System (CIIS), UiTM has 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 six main modules encompassing the complete Student Life Cycle process, from Intake to Convocation and Alumni. The four 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.

We have been fortunate to acquire most of our examination data from the CIIS. The data are classified by faculty and branch campuses. The total number

**Table 1.** Characteristics of UiTM dataset

| 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 |

of students enrolled in Semester 2, 2002/03 was 84,675, course enrolment was 472,201 and the total number of courses to be scheduled was 2,650 (see Table 1).

The data supplied from CIIS were in the form of: a list of examinations that must be scheduled, a list of examinations 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 examinations has 2,650 examinations to be scheduled, but after computing the enrolment for each examination using the student file, examinations with zero enrolments were removed 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. There are 491 examinations with zero enrolments and so we only need to schedule 2,159 examinations. Data accuracy is one of the difficulties in processing data from a large university that involves many faculties and campuses. We believe that these examinations have zero enrolments because officers in faculties and campuses may not have up-to-date information on the exact course enrolments and, also, students add or drop courses at the last moment.

### 2.2 Problem Formulation

We can represent the examination timetabling problem as follows:

1. $E$: a set of $m$ examinations $E_1$, $E_2, \ldots, E_m$.
2. $S$: a set of $n$ slots $\mathrm{S}_1$, $\mathrm{S}_2, \ldots, S_n$.

3. $U$: a set of $u$ campuses $\mathrm{U}_1$, $\mathrm{U}_2, \ldots, U_u$.
4. A final examination timetable $T_{mn}$ such that: $T_{ik} = 1$ if examination $i$ is scheduled in slot $k$, 0 otherwise.
5. 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.
6. A conflict matrix $C_{mm}$ such that $C_{ij}$ = total number of students sitting for both examinations $i$ and $j$ categorised by campus type.
7. A co-schedule matrix $R_{mm}$ such that $R_{ik} = 1$ if examination $i$ and examination $k$ 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 the examination scheduler must schedule examinations into at most $n$ slots. There are 2,159 examinations with some examinations being held concurrently. Even though each examination may have a different duration (120, 150 or 180 minutes), we will treat each examination as occupying a complete slot of 180 minutes. The total number of examination days is 20 and each day will have two slots (morning and afternoon), i.e. we will have 40 slots in which to assign examinations. We categorise each campus (A or B) to indicate which days or slots must not be used in assigning examinations taken by students in that particular campus. Campuses in category A have weekend: half-day on Saturday and full-day on Sunday while campuses in category B have weekend: half-day on Thursday and full-day on Friday. The examination dates were from 20th April 2003 to 10th May 2003 with 1st May as a public holiday across all campuses, so no examinations 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.

**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 examination in the same slot:

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

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

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

For all examinations $i$:

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

All examinations that must be scheduled with examination $i$ should be assigned to the same slot $k$.

c All examinations must be scheduled:

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

Each examination, $(E_1, E_2, \ldots, E_m)$ should be scheduled only once.

**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 examination $i$ 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 examinations during the weekend. We can try to produce a better quality timetable by penalising examinations that are scheduled during the weekend. Weekends for category A campuses are half-day on Saturday and full-day on Sunday and weekends for category B campuses are 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 examination. $W_{ik}$ is the proximity cost if examination $i$ 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:

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

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

## 3 Tabu Search Based Hyper-heuristic

A hyper-heuristic [7], [9] works at a higher level of abstraction than a meta-heuristic and does not necessarily 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 details and only returns a new solution and a revised value for the 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
             Choose the best heuristic (with the best improvement)
             Apply chosen heuristic and make the heuristic tabu
             Update solution
             Update the tabu status of heuristics in the tabu list
         Until terminating condition
```

The initial solution is produced using a constructive heuristic (largest degree or saturation degree [17]. Next, a randomisation (randomly move examinations 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 are candidates to be applied and which heuristic we actually apply. Each heuristic differs in how it decides to move, thus creating its own search

region (heuristic search space) in the solution search space. In the search for good quality solutions, the hyper-heuristic exhibits a type 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 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 compared the final best solution. In our previous work [27], we showed empirically, using deterministic tabu durations, 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 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 durations and random dynamic tabu durations. Random dynamic tabu duration uses a tabu duration range ($t_{min}$ and $t_{max}$), and at each decision point, when 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. At each iteration, we compare the solution from each heuristic and take the best solution. We use three different strategies for accepting the best 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 [22] in his great deluge algorithm (GDA). Burke et al. [13] have also applied it to examination timetabling. The algorithm deals with feasible solutions and will accept worse solutions if the cost evaluation is less than or equal to a boundary, called level. The level is reduced at each iteration using a decay rate that is dependent upon the initial cost evaluation, desired cost evaluation and the time for the algorithm to achieve this desired cost.

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

1. The simplest form, i.e. the Basic Tabu Search Hyper-heuristic (TSHH-B), considers all tabu inactive heuristics and applies the heuristic that has the best improvement only. The algorithm iterates for a fixed time (4 hours).
2. The second hyper-heuristic is Tabu Search Hyper-heuristic with Hill Climbing (TSHH-HC), which adds to TSHH-B a successive call to the best performing heuristic until no further improvement is made.
3. The third hyper-heuristic is Tabu Search Hyper-heuristic with Great Deluge (TSHH-GD), which updates a solution within a certain boundary only. The execution time is limited by a number of steps (i.e. 10,000,000 iterations) and the algorithm will terminate if there is no improvement in the last 10,000 iterations. The time taken by a TSHH-GD iteration is longer compared to GDA because GDA generates only one new solution by doing a random move while TSHH-GD generates several solutions as it calls every heuristic which is not tabu. Each low-level heuristic may also take a longer time than just a random move. The decay rate is calculated as the difference between the desired cost and the initial cost divided by the number of iterations. In each iteration, the boundary is lowered by the decay rate. A solution from the best performing low-level heuristic is accepted to modify the current solution if it is better than the previous solution or it is lower than the boundary.

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

### 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 examination into an unscheduled list (examination becomes unscheduled and violates hard constraints).

We use the same low-level heuristics as in [27]:

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

**Table 2.** Comparison of results between Uitm-03 and Car-s-91 datasets

| Tabu | Uitm-03 | | | Car-s-91 | | |
|---|---|---|---|---|---|---|
| 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 | 1.40 | 5.43 | 5.31 | 5.6 |

## 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 [27], 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 solutions 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 the tabu search approach of Di Gaspero and Schaerf [21]. 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 (TSHH-B) and proximity cost evaluation function on our new dataset. We also run our new and improved method: tabu-search-based hyper-heuristic with hill climbing (TSHH-HC) and tabu-search-based hyper-heuristic with great deluge (TSHH-GD), with Carter's examination timetabling benchmark data and the Uitm-03 dataset. Here, we discuss the algorithm performance when applied to the Uitm-03 dataset and compare its behaviour with one other dataset, i.e. Car-s-91 (one of the benchmark datasets). Car-s-91 (Carleton University, Ottawa) is one of the largest datasets with 682 examinations to be scheduled in 35 slots, 16,925 students and 56,877 student examinations.

The first column in Table 2 shows the tabu duration, i.e. how long a heuristic is placed in the tabu list. We tested with both deterministic and random dynamic tabu durations. The best result for Car-s-91 is 4.50 published by Yong and

**Table 3.** Compare results for UiTM dataset with other solution methods

| | TSHH-B | TSHH-HC | TSHH-GD | Manual | GD | SA |
|---|---|---|---|---|---|---|
| UiTM-03 | 1.55 | 1.55 | 1.35 | 12.83 | 1.40 | 1.68 |

Petrovic [38]. They used a combination of case base reasoning using a fuzzy similarity measure to choose sequential heuristics during the construction phase and then the great deluge algorithm. The next best result of 4.65, published by Burke and Newall [1], uses a combination of adaptive initialisation strategies and the great deluge algorithm. Both methods are based upon the idea that good initial solutions fed to the great deluge algorithm will produce good solutions. Referring to Table 2, our best result for Car-s-91 is 4.93 using the tabu-search-based hyper-heuristic and the great deluge with a tabu duration of 3. Thus, the hyper-heuristic is able to produce a good-quality solution for the Car-s-91 dataset, even though the initial solution is not as good as the initial solutions in [1], [38]. Our method shows similar behaviour between both datasets, where the best result is obtained using tabu search hyper-heuristic and great deluge with a tabu duration of 3. Table 3 compares our best result on the UiTM dataset with the existing manual solution, the great deluge algorithm and simulated annealing.

Figures 1 and 2 show the best proximity cost, with various tabu durations obtained for the 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 return 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 given problem instance. For the UiTM dataset (Figure 1), as the tabu duration increases, solution quality improves, and once it reaches its best tabu duration, the solution quality begins to deteriorate as we increase the tabu duration further. Car-s-91 (Figure 2) shows only a slight difference when the 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 the tabu duration is one and improves again after that.

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-B 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

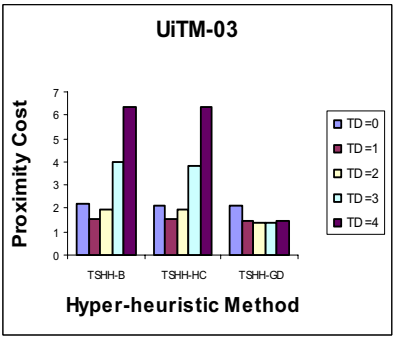


**Fig. 1.** Hyper-heuristic methods on UiTM-03

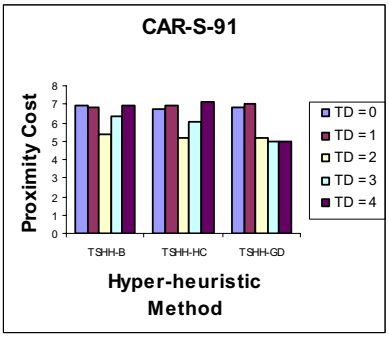


**Fig. 2.** Hyper-heuristic methods on Car-s-91

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. As we increase the tabu duration, it shows more movement and exploration of search space and is therefore able to find a better quality 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 (see the algorithm in [11]). We ran both algorithms for 4 hours and traced 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 explores various heuristic solution space and may take 10 times longer compared to the great deluge algorithm.

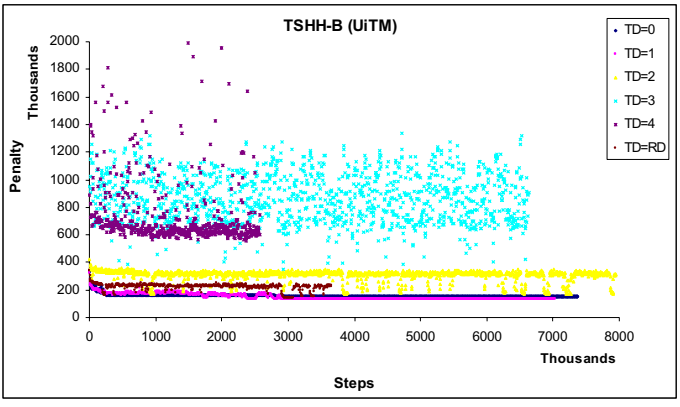


**Fig. 3.** Basic tabu-search-based hyper-heuristic for UiTM dataset

### 4.2 Proximity and Weekend Cost Evaluation Method

The UiTM dataset has an additional constraint or criterion, i.e. a weekend cost that needs to be minimised. The proximity cost and weekend cost are combined into 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 examination.

The high-level part of the hyper-heuristic does not need to be changed. The low-level heuristics, which require more knowledge about the problem, need to be adjusted. The cost evaluation function is also modified to reflect a change in the objective function. The unscheduled examination weight needs adjustment (new weight of 10,000) and a new weekend weight added. In the previous implementation, two heuristics minimise the second- and third-order conflict (a conflict by a student having to sit for examinations, 1 or 2 slots apart). In the current implementation, a new heuristic, reduces a conflict by a student who has to sit for a weekend examination.

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. A separate low-level heuristic can be used to minimise a specific criteria and the hyper-heuristic can choose which criteria to optimise in each decision step. In this way, the tabu search hyper-heuristic is also a general method that can solve an examination timetabling problem with multiple objectives. Petrovic and Bykov [31] solve the examination timetabling problem with multiple objectives by dynamically changing the weights of the criterion during the search process. The hyper-heuristic approach does this implicitly because it has the intelligence to choose between the low-level heuristic that minimises the first or the second criterion. The only parameter that we need to adjust would be the most suitable weights associated with the conflicts or criteria.

In the UiTM problem, it is important to associate appropriate weights with the weekend conflict violation and the unscheduled examination. Currently the

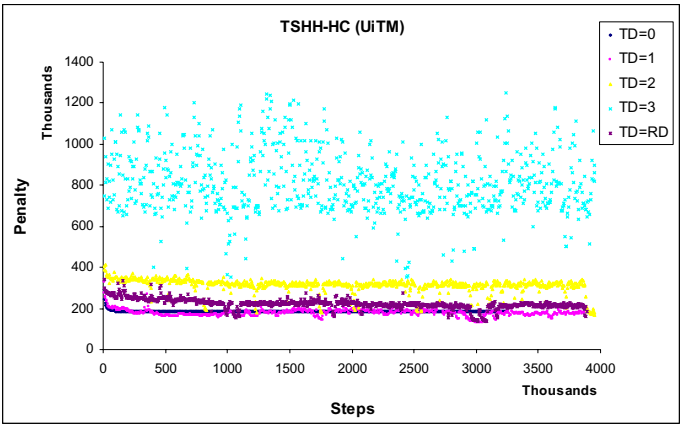


**Fig. 4.** Tabu-search-based hyper-heuristic with Hill Climbing for UiTM dataset

low-level heuristics are able to optimise a bi-criteria problem and can easily be extended to optimise a number of criteria (e.g. minimise usage of smaller rooms). In deciding what is an appropriate weight, we need to consider the importance of a criterion and how it will influence the search direction. A high weekend weight will push the search into a higher proximity cost since the number of examination slots will be reduced. Experiments were conducted with weights of 32 and 16 associated with a weekend conflict. Finally, a weight of 16 is chosen because a weight of 32 was too high and caused the search to become trapped and not able to improve the solution. The weekend weight is also the same (and therefore as important) as the weight for students who have examinations in adjacent periods.

The unscheduled examination weight of 5,000 that is being used with benchmark datasets is also not suitable for the UiTM dataset. Normally the unscheduled weight determines whether an examination is better being unscheduled, or scheduled and causing high proximity cost. A weight of 5,000 for an unscheduled examination 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 examination heuristic will outperform other heuristics by un-scheduling an examination. In the UiTM problem, the unscheduled examination weight is fixed at 10,000.

**UiTM Proximity and Weekend Cost Analysis.** Table 4 shows the same hyper-heuristics being applied but with the different objective function. The results should be higher than in Table 2 since we are also taking into account the penalty for weekend examinations. On average, we could say that students may not have to sit for weekend examinations or sit for an adjacent examination since the best solution (cost of 7.12 by TSHH-GD with tabu duration = 3) is less than the weight for a weekend examination or adjacent examinations. However, 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 or weekend examinations.

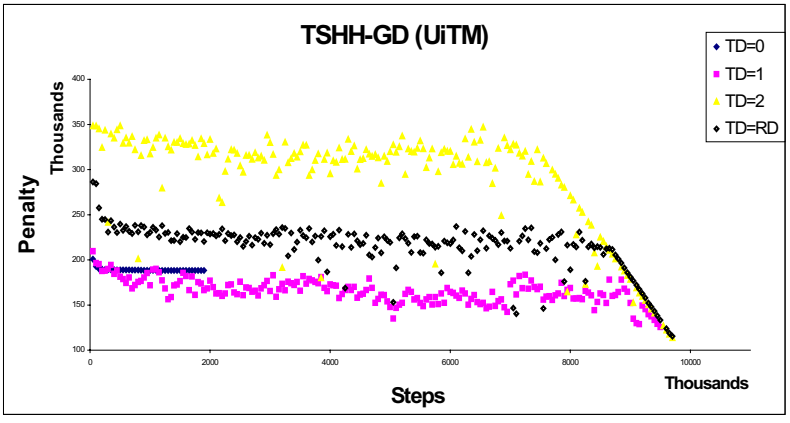


**Fig. 5.** Tabu-search-based hyper-heuristic with Great Deluge for UiTM dataset

**Table 4.** UiTM-03 dataset that minimised proximity and weekend cost

| Tabu | UiTM-03 | | |
|---|---|---|---|
| Duration | TSHH-S | 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 |

Tables 5 and 6 show the number of students and percentage of students involved in a conflict that occur in two of the solutions produced. Table 5 is an analysis of the best result by the tabu search based hyper-heuristic with great deluge (TD = 3). Table 6 is an analysis of the best result by tabu search based hyper-heuristic with hill climbing (TD = 1). The columns show each of the constraints that we are minimising. The gap number indicates a gap between two examinations that a student is sitting. The results seem consistent with the objective of minimising proximity and weekend cost. The best solution has a lower cost in all of the individual constraints that we wish to minimise.

The weekend cost contributes a high percentage to the total cost in both solutions (last row in Tables 5 and 6). Therefore, it is beneficial to look at how the initial solution and the unscheduled examination weight affect the performance of the hyper-heuristic.

**New Initial Solution and Dynamic Unscheduled Examination Weight.** Previously, the constructive heuristic used to generate the initial solution for the

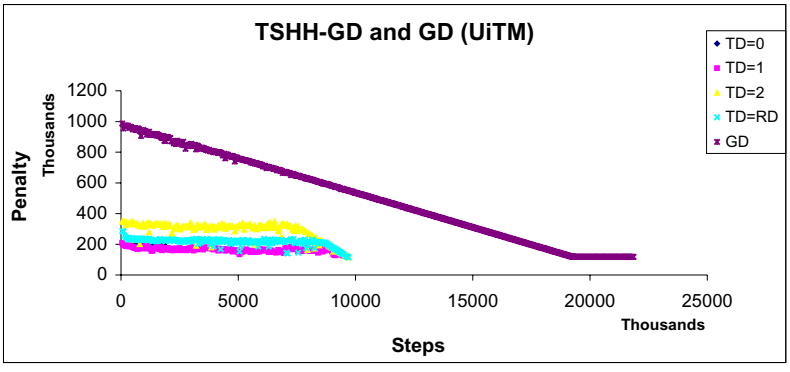


**Fig. 6.** Comparing TSHH-GD and Great Deluge algorithm performance

**Table 5.** Data characteristic of solution with 7.12 proximity and weekend cost (by TSHH-GD with TD = 3)

| | Gap = 1 | Gap = 2 | Gap = 3 | Gap = 4 | Gap = 5 | Weekend |
|---|---|---|---|---|---|---|
| Total cost | 88,464 | 86,536 | 59,440 | 45,104 | 29,488 | 294,208 |
| Avg cost per student | 1.04 | 1.02 | 0.70 | 0.53 | 0.35 | 3.47 |
| Number of students | 5,529 | 10,817 | 14,860 | 22,552 | 29,488 | 18,388 |
| % of total students | 6.53% | 12.77% | 17.55% | 26.63% | 34.82% | 21.72% |
| % of cost | 14.66% | 14.35% | 9.85% | 7.48% | 4.89% | 48.77% |

Uitm-03 dataset was the same as the constructive heuristic for the benchmark datasets. With that initial solution, the hyper-heuristic is able to produce a feasible solution that is much better when compared to the existing manual solution. The constructive heuristic for benchmark datasets does not need to differentiate between weekend and non-weekend slots, and therefore does not prioritise which slots to assign an examination to. The results for Uitm-03 dataset in Tables 5 and 6 show that the weekend cost contributes almost half of the total cost. Therefore, it is worth considering prioritising non-weekend slots in the constructive heuristic, i.e. we need to inject more knowledge about the problem domain so as to produce an initial solution that starts from a reasonable point in the search space that does not violate one of the important constraints that we want to minimise. The only modification in the constructive heuristic is in determining which is the first available slot for an examination. The new constructive algorithm will only consider weekend slots if non-weekend slots are unavailable. During the search, the algorithm will also use the same criteria when moving an examination. This method should direct the search along a trajectory that avoids weekend slots.

Another issue that needs to be addressed is whether the unscheduled weight of 10,000 is suitable for the Uitm-03 dataset. In deciding which low-level heuristic

**Table 6.** Data characteristic of solution with 9.27 proximity and weekend cost (by TSHH-B with TD = 1)

| | Gap = 1 | Gap = 2 | Gap = 3 | Gap = 4 | Gap = 5 | Weekend |
|---|---|---|---|---|---|---|
| Total cost | 172,688 | 115,512 | 69,344 | 53,930 | 30,579 | 343,296 |
| Avg cost per student | 2.04 | 1.36 | 0.82 | 0.64 | 0.36 | 4.05 |
| Number of students | 10,793 | 14,439 | 17,336 | 26,965 | 30,579 | 21,456 |
| % of total students | 12.75% | 17.05% | 20.47% | 31.85% | 36.11% | 25.34% |
| % of cost | 21.99% | 14.71% | 8.83% | 6.87% | 3.89% | 43.71% |

to select, the hyper-heuristic uses two important factors, i.e. cost of the objective function and non-tabu low-level heuristics. If the *remove examination* heuristic un-schedules an examination with a proximity cost higher than the unscheduled weight, it would seem advantageous to un-schedule the examination. The heuristics that reschedule examinations might find it difficult to improve the objective function since the new proximity cost may be greater than the un-schedule cost. This will eventually cause some examinations not being scheduled at the end of the search. The unscheduled weight does play an important role in deciding whether we need to un-schedule some examinations so that other conflicting examinations can be scheduled into the vacated slot. Therefore, instead of using a fixed unscheduled weight, we can use a dynamic un-scheduled weight that is dependent upon the number of students. Thus, in this implementation, the cost of un-scheduling an examination differs for each examination, i.e. the cost of un-scheduling an examination $e_i$ is calculated as 60 multiplied by the number of students enrolled in examination $e_i$ or 5,000, whichever is the maximum.

Table 7 shows a comparison of the results obtained using two different initial solutions and fixed and dynamic unscheduled weights. The old solution indicates an initial solution used in the previous experiment (Table 4). The new solution is produced using the new constructive heuristic. The last four columns in Table 7 show the results produced by TSHH-B and TSHH-HC with fixed deterministic tabu duration between 0 and 4. Each run takes one hour. Since we already have results from previous experiment (Table 4) of a four-hour run with fixed unscheduled weight and old initial solution, we did not run it again for one hour. Moreover, the one-hour run with the new parameters has already improved the results compared to the four-hour run. The TSHH-GD is not tested with this parameter because we want to observe how the dynamic unscheduled weight and new initial solution affect the final results and it should be sufficient to test the first two methods only.

From Table 7, we can conclude that with the new solution, both methods are able to produce better solutions compared to the best method (TSHH-GD) that was initialised with the old solution (Table 4). A dynamic unscheduled weight also assists the hyper-heuristic in deciding which low-level heuristic is in favour. Since the best solution is when TD equals zero, this implies that the hyper-heuristic does not even need a tabu list to help manage the low-level heuristics.

**Table 7.** Results with two different initial solutions and two unscheduled weight mode

| | | Dynamic weight | | Fixed weight | |
|---|---|---|---|---|---|
| | TD | Old solution (1 h run) | New solution (1 h run) | Old solution (4 h run)* | New solution (1 h run) |
| Basic tabu search based hyper-heuristic (TSHH-B) | 0 | **6.98** | **4.40** | 10.10 | 18.43 |
| | 1 | 7.67 | 4.91 | **9.27** | 20.60 |
| | 2 | 10.23 | 6.64 | 9.88 | **7.16** |
| | 3 | 16.99 | 8.73 | 17.36 | 8.67 |
| | 4 | 16.75 | 15.11 | 16.42 | 15.43 |
| Tabu search based hyper-heuristic with hill climbing (TSHH-HC) | 0 | **6.77** | **4.25** | **9.29** | 16.64 |
| | 1 | 7.70 | 6.01 | 11.20 | 16.88 |
| | 2 | 10.12 | 5.93 | 9.70 | **6.94** |
| | 3 | 16.91 | 8.95 | 17.31 | 9.94 |
| | 4 | 16.30 | 14.74 | 17.16 | 16.09 |

* As Table 4.

**Table 8.** Data characteristic of solution with 4.40 proximity and weekend cost. Best solution for TSHH-B with new initial solution and dynamic weight.

| | Gap = 1 | Gap = 2 | Gap = 3 | Gap = 4 | Gap = 5 | Weekend |
|---|---|---|---|---|---|---|
| Total cost | 80,672 | 73,984 | 58,820 | 50,476 | 30,881 | 78,064 |
| Avg cost per student | 0.95 | 0.87 | 0.69 | 0.60 | 0.36 | 0.92 |
| Number of students | 5,042 | 9,248 | 14,705 | 25,238 | 30,881 | 4,879 |
| % of total students | 5.95% | 10.92% | 17.37% | 29.81% | 36.47% | 5.76% |
| % of cost | 21.63% | 19.84% | 15.77% | 13.54% | 8.28% | 20.93% |

**Table 9.** Data characteristic of solution with 6.98 proximity and weekend cost. Best solution for TSHH-B with old initial solution and dynamic weight.

| | Gap = 1 | Gap = 2 | Gap = 3 | Gap = 4 | Gap = 5 | Weekend |
|---|---|---|---|---|---|---|
| Total cost | 91,360 | 89,440 | 55,808 | 47,600 | 29,472 | 277,440 |
| Avg cost per student | 1.08 | 1.06 | 0.66 | 0.56 | 0.35 | 3.28 |
| Number of students | 5,710 | 11,180 | 13,952 | 23,800 | 29,472 | 17,340 |
| % of total students | 6.74% | 13.20% | 16.48% | 28.11% | 34.81% | 20.48% |
| % of cost | 15.46% | 15.13% | 9.44% | 8.05% | 4.99% | 46.93% |

Tables 8–10 present a detailed analysis of the best solution with different parameters. With the new initial solution that avoids weekend slots, the hyper-heuristic is able to improve the weekend conflicts, implying that with more knowledge about the problem domain injected into the initial solution construction phase and the low-level heuristics, the hyper-heuristic can perform better.

**Table 10.** Data characteristic of solution with 6.94 proximity and weekend cost. Best solution for TSHH-HC with new initial solution and fixed weight.

| | Gap = 1 | Gap = 2 | Gap = 3 | Gap = 4 | Gap = 5 | Weekend |
|---|---|---|---|---|---|---|
| Total cost | 135,360 | 92,664 | 89,080 | 58,090 | 32,723 | 89,768 |
| Avg cost per student | 1.60 | 1.09 | 1.05 | 0.69 | 0.39 | 1.06 |
| Number of students | 8,460 | 11,583 | 22,270 | 29,045 | 32,723 | 5,611 |
| % of total students | 9.99% | 13.68% | 26.30% | 34.30% | 38.65% | 6.63% |
| % of cost | 27.20% | 18.62% | 17.90% | 11.67% | 6.58% | 18.04% |

With this good initial solution and a fixed unscheduled weight, we can also get a reasonable result compared to solutions with a dynamic unscheduled weight. As a general method and without the extra knowledge, the hyper-heuristic can also produce competitive results (compared to a manual solution) for the UiTM dataset specification (as shown in Table 3).

## 5 Conclusions

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 schedules that are feasible and may satisfy everybody involved. The Uitm-03 dataset is a large real-world examination timetabling problem. Its data may not exactly reflect all the required information such as validated student data and implicit constraints, however, other crucial data such as the actual timetable used and the list of examinations to be scheduled are consistent with the real problem. All data inconsistencies were removed prior to processing so that the dataset is as close as possible to the actual problem. The important constraints are considered in the search engine with a flexibility of adding additional constraints if necessary.

Tabu-search-based hyper-heuristics have been shown to produce feasible and good quality solutions on other benchmark datasets [27]. The same hyper-heuristic framework was tested with the Uitm-03 dataset. An additional criterion of minimising the weekend cost is added to the objective function and with no modification to the hyper-heuristic framework it is able to produce good quality solutions. It is obvious that the nature of the Uitm-03 dataset is different from the benchmark datasets because of an additional objective criteria, a larger number of students and the maximum number of examinations in conflict is high compared to the average number of examinations in conflict. Thus, by adding such knowledge in the form of a new constructive heuristic in the initialisation phase, prioritising non-weekend slots and dynamic unscheduled examination weights, the hyper-heuristic framework is able to produce a much better solution. It is such knowledge that differentiates problem instances and as such the hyper-heuristic is a general method that was shown to be effective on benchmark datasets and also on the UiTM problem.

Currently, the tabu search hyper-heuristic method is dependent upon the tabu duration value. Future work will investigate an adaptive tabu strategy of selecting the best tabu duration at each decision point. This method should free the hyper-heuristic from the problems of parameter tuning.

## Acknowledgements

This research was carried out at the University of Nottingham and supported by the Malaysian Public Services Department and MARA University of Technology (UiTM), Malaysia.

## References

1. Burke, E. K., Newall, J. P.: Solving Examination Timetabling Problems Through Adaption of Heuristic Orderings. Ann. Oper. Res. **129** (2004) 107–134
2. Burke, E. K., Newall, J. P.: Enhancing Timetable Solutions with Local Search Methods. In: Burke, E., De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling IV (PATAT'02, Selected Papers). Lecture Notes in Computer Science, Vol. 2740. Springer, Berlin (2003) 195–206
3. Burke, E. K., Petrovic, S.: Recent Research Directions in Automated Timetabling. Eur. J. Oper. Res. **140** (2002) 266–280
4. Burke E. K., Landa Silva, J. D., Soubeiga, E.: Multi-objective Hyper-heuristic Approaches for Space Allocation and Timetabling. In: Ibaraki T., Nonobe K., Yagiura M. (eds.): Meta-heuristics: Progress as Real Problem Solvers. Springer, Berlin (2005)
5. Burke, E. K., Meisels, A., Petrovic, S., Qu, R.,A: Graph-Based Hyper-heuristic for Timetabling Problems. Eur. J. Oper. Res. (2005) accepted for publication
6. Burke, E. K., Petrovic, S. and Qu, R.: Case Based Heuristic Selection for Timetabling Problems. J. Scheduling (2005) accepted for publication
7. Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-Heuristics: An Emerging Direction in Modern Search Technology. Chapter 16 in: Glover, F., Kochenberger, G. (eds.): Handbook of Meta-Heuristics. Kluwer, Dordrecht (2003) 457–474
8. Burke, E. K., Jackson, K. S., Kingston, J. H., Weare, R. F.: Automated Timetabling: The State of the Art. Comput. J. **. 40** (1997) 565–571
9. Burke, E. K., Kendall, G., Soubeiga, E. (2003b) A Tabu-Search Hyperheuristic for Timetabling and Rostering. J. Heuristics **9** (2003) 451–470
10. Burke, E. K., Elliman, D. G., Ford, P. H., Weare, R. F.: Examination Timetabling in British Universities—A Survey. In: Burke, E., Ross, P. (eds.): The Practice and Theory of Automated Timetabling I (PATAT'95, Selected Papers). Lecture Notes in Computer Science, Vol. 1153, Springer, Berlin (1996) 76–92
11. Burke, E. K., Newall, J. P., Weare, R.: Initialisation Strategies and Diversity in Evolutionary Timetabling. IEEE Trans. on Evol. Comput. **6** (1998) 81–103
12. Burke, E. K., Newall, J. P., Weare, R. F.: A Memetic Algorithm for University Exam Timetabling. In: Burke, E., Ross, P. (eds.): PATAT 1995—Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling 496–503

13. Burke, E. K., Bykov, Y., Newall, J., Petrovic, S.: A Time-Predefined Local Search Approach to Exam Timetabling Problems. IIE Trans. on Oper. Eng. **36** (2004) 509–528
14. Caprara, A., Fischetti, M., Guida, P. L., Monaci, M., Sacco, G., Toth, P.: Solution of Real-World Train Timetabling Problems. Proceedings of the 34th Annual Hawaii International Conference on System Sciences (2001) 1057–1066
15. Carter, M. W.: A Survey of Practical Applications of Examination Timetabling Algorithms. Oper. Res. Soc. Am. **34** (1986) 193–202
16. Carter, M. W.: EXAMINE: A General Examination Timetabling System. In: Burke, E. K., Carter, M. (eds.): PATAT 1997—Proceedings of the 2nd International Conference on the Practice and Theory of Automated Timetabling 363
17. Carter, M. W., Laporte, G.: Recent Developments in Practical Examination Timetabling. In: Burke, E., Ross, P. (eds.): The Practice and Theory of Automated Timetabling I (PATAT'95, Selected Papers). Lecture Notes in Computer Science, Vol. 1153, Springer, Berlin (1996) 3–21
18. Carter, M. W., Laporte, G., Lee, S. Y.: Examination Timetabling: Algorithmic Strategies and Applications. J. Oper. Res. Soc. **47** (1996) 373–383
19. Comm, C. L., Mathaisel, D. F. X.: College Course Scheduling. A Market for Computer Software Support. J. Res. Comput. Educ. **21** (1988) 187–195
20. Cowling, P., Kendall, G., Mohd Hussin, N.: A Survey and Case Study of Practical Examination Timetabling Problems. In: Burke, E., De Causmaecker, P. (eds.): PATAT 2002—Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling 258–261
21. Di Gaspero, L., Schaerf, A.: A Tabu Search Techniques for Examination Timetabling. In: Burke, E., Erben, W. (eds.): The Practice and Theory of Automated Timetabling III (PATAT'00, Selected Papers). Lecture Notes in Computer Science, Vol. 2079. Springer, Berlin (2001) 104–117
22. Dueck, G.: New Optimization Heuristics for the Great Deluge Algorithm and the Record-to-Record Travel. J. Comput. Phys. **104** (1993) 86–92
23. Forster, G.: Syllabus Plus: A State-of-the-Art Planning and Scheduling System for Universities and Colleges. In: Burke, E., Ross, P. (eds.): PATAT 1995—Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling 244–252
24. Goltz, H.-J., Matzke D.: ConBaTT—Constraint-Based Timetabling. In: Burke, E. K., Erben, W. (eds.): PATAT 2000—Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling 491
25. Isaai, M. T., Singh, M. G.: Hybrid Applications of Constraint Satisfaction and Meta-heuristics to Railway Timetabling: A Comparative Study. IEEE Trans. on Systems, Man and Cybernetics, Part C: Applications and Reviews. **31** (2001) 87–95
26. JTAP: Central Timetabling By Computer: A review of Existing Information. Report by JISC Technology Applications Programme, June (1998). The report can be downloaded from url:http://www.jisc.ac.uk/uploaded_document/jtap-021.doc
27. Kendall G., Mohd Hussin N.: An Investigation of a Tabu Search Based Hyperheuristic for Examination Timetabling. In: Kendall, G.. et al. (eds.): Multidisciplinary Scheduling: Theory and Applications. Springer, Berlin (2005) 309–328. An extended abstract of this paper also appeared in the Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA'03) 226–233

28. McCollum, B., Newall, J.: Introducing Optime: Examination Timetabling Software. In: Burke, E. K., Erben, W. (eds.): PATAT 2000—Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling 485
29. Merlot, L. T. G., Boland, N, Hughes, B. D., Stuckey, P. J.: A Hybrid Algorithm for the Examination Timetabling Problem. In: Burke, E., De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling IV (PATAT'02, Selected Papers). Lecture Notes in Computer Science, Vol. 2740. Springer, Berlin (2003) 207–231
30. Petrovic, S., Burke, E. K.: University Timetabling. Chapter 45 in: Leung, J. (ed.): The Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Boca Raton, FL (2004)
31. Petrovic, S., Bykov, Y.: A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories. In: Burke, E., De Causmaecker, P. (eds.): Practice and Theory of Automated Timetabling IV (PATAT'02, Selected Papers). Lecture Notes in Computer Science, Vol. 2740. Springer, Berlin (2003) 181–194
32. Rogalla, S.: CELCAT: A Practical Solution to Scheduling Problems, Corbett Engineering, UK. In: Proceedings of The Practice and Theory of Automated Timetabling II (PATAT'98) 368
33. Schaerf, A.: A Survey of Automated Timetabling. Artif. Intell. Rev. **13** (1999) 87–127
34. Terashima-Marín, H., Ross, P. M., Valenzuela-Rendón, M.: Evolution of Constraint Satisfaction Strategies in Examination Timetabling. In: Proceedings of the Genetic and Evolutionary Conference (1999) 635–642
35. Trick, M. A.: A Schedule-then-Break Approach to Sports Timetabling. In: Burke, E., Erben, W. (eds.): The Practice and Theory of Automated Timetabling III (PATAT'00, Selected Papers). Lecture Notes in Computer Science, Vol. 2079. Springer, Berlin (2001) 242–253
36. Wan Ya, Baharudin, N. Interview with Manager and System Analyst. Examination Unit, Center for Integrated Information System, MARA University of Technology, August 2001
37. White, G. M., Xie, B. S.: Examination Timetables and Tabu Search with Longer Term Memory. In: Burke, E., Erben, W. (eds.): The Practice and Theory of Automated Timetabling III (PATAT'00, Selected Papers). Lecture Notes in Computer Science, Vol. 2079. Springer, Berlin (2001) 85–103
38. Yang, Y., Petrovic, S.: A Novel Similarity Measure for Heuristic Selection in Examination Timetabling. In: Burke, E. K., Trick, M. (eds.): Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (2004) 377–396