ELSEVIER

# Linear Combination of Heuristic Orderings in Constructing Examination Timetable

Syariza Abdul-Rahman[a], Andrzej Bargiela[a], Edmund K. Burke[a], Ender Özcan[a], Barry McCollum[b]

[a]*University of Nottingham, School of Computer Science, Jubilee Campus, Nottingham NG8 1BB, UK*
[b]*Queen's University Belfast, School of Electronics, Electrical Engineering and Computer Science, University Road, Belfast, BT7 1NN, Northern Ireland, UK*

## Abstract

In this paper, we investigate linear combinations of graph coloring heuristics with a heuristic modifier for solving the examination timetabling problem. We invoke a normalisation strategy for each parameter in order to generalise the specific problem data. Two graph coloring heuristics were used in this study (largest degree and saturation degree). A score for the difficulty of assigning each examination was obtained from a linear combination of these two heuristics and examinations in the list were ordered based on this value. The examinations with the higher difficulty_score value were chosen for scheduling based on two strategies. We have tested for single and multiple heuristics with and without a heuristic modifier with different combinations of weight values for each parameter on the Toronto and ITC2007 benchmark data sets. We have observed that the combination of multiple heuristics with a heuristic modifier offers the most effective way to obtain good solution quality. Experimental results showed that our approach has outperformed other constructive approaches on one instance of Toronto benchmark data sets. We conclude that this linear combination of heuristics is a highly effective method and simple to implement.

*Keywords:* constructive heuristic, linear combination, graph colouring

## 1. Introduction

The examination timetabling problem has been much studied and a wide variety of approaches have been taken across a variety of associated problem descriptions. In general, the task is NP hard [1]. The real world problem is rich and varied, involving significant levels of information. It has become increasingly difficult in recent years due to the increasing size of student enrolments and different choices of courses ([2]). The manual solution of this problem is typically suboptimal (feasible but not a very good solution) since high quality exploration of the space is beyond the scope of ad-hoc search. Examination timetabling problems have been well documented in the academic literature with a good coverage of various methods and strategies to solve this problem ([3], [4], [5]).

The examination timetabling problem can be defined as the assignment of a finite set of examinations to a finite set of time-slots while, at the same time, satisfying various problem constraints. It involves two types of constraints; hard constraints and soft constraints. The hard constraints are strictly required to be adhered to in any circumstances. Satisfying the hard constraints produces a *feasible* solution. For example, students cannot sit two examinations at the same time. On the other hand, soft constraints do not affect the feasibility of the solution but they need to be satisfied as much as possible for the solution to be of high quality. Of course, soft constraints usually have to be violated to some degree in a real world situation. The extent to which the defined soft constraints are satisfied reflects the quality of the obtained timetable. An example of a soft constraint is that students should have as much time between

examinations as possible. Based on a survey ([6]), there are many different such constraints that have been highlighted by different academic institutions in Britain.

The examination timetabling problem can be mapped through an identity relationship onto a graph colouring mathematical formalism. Indeed, this observation underpins some of the earliest and most well known approaches to examination timetabling problems ([3]). In the graph colouring formalism, the vertices represent examinations and the edges connecting vertices represent hard constraint i.e. conflicts between the examinations. For more details on graph representation in timetabling problem see [7].

Timetabling approaches have been widely investigated at the interface of artificial intelligent and operation research over the last decades or so. One of the earliest approaches in educational timetabling is graph colouring approaches that are designed based on graph theory ([3]). Many other approaches upon meta-heuristic technique and hybridization have shown great success such as evolutionary algorithm ([8], [9], [10]), tabu search ([11]), ant algorithm ([12]) and simulated annealing ([13], [14]). These approaches aim to improve the initial solutions by employing search strategy to escape from local optima.

Recently, variants of local search approaches are widely used to solve timetabling problem. These approaches work by navigating the solution search space and exploring neighbourhood structures in a different way from meta-heuristics. These local search approaches include very large neighbourhood search ([15]), variable neighbourhood search ([16]), iterated local search ([17]) and GRASP ([18]). The reliant on parameter setting of these approaches has lead to the introduction of other approaches such as hyper-heuristic ([19]), case-based reasoning ([20]), fuzzy approaches ([21]), constraint based approaches ([22]) and granular information processing ([23]) within the timetabling arena. A review on major approaches in examination timetabling can be found in [3], [4], [24], [7], [25] and [5].

The successful assignment of an examination to a time-slot is closely related to the initial ordering strategy in which all examinations are processed. Consequently, examinations are first ordered according to the perceived difficulty of being scheduled in the available time-slots. The examinations are then taken from the order and assigned one by one to the time-slot. The examination deemed to be the most difficult is scheduled first in the timetable in the hope that the remaining scheduling problem is less difficult than the original one and at the same time the relatively least difficult examinations will have less assignment trouble later on in the process. The approaches to timetabling which encompass this basic graph colouring implementation are called constructive approaches and are often used during the initialisation strategy of a meta-heuristic process. In the past, there have been various ordering strategies employed in the context of examination timetabling ([3], [7]). Commonly used ordering strategies are: saturation degree, largest degree, largest weighted degree, largest enrolment and colour degree. Generally, hyper-heuristic is one of approaches that used graph colouring heuristics as low-level heuristics to construct solution. This approach works as a high-level heuristic and intelligently chooses a set of low-level heuristics based on learning mechanism ([26], [27]).

Since none of the ordering strategies provides a guarantee of successful scheduling, there has been a wide study on ordering heuristics within adaptive approaches reported in the academic literature. In our previous study ([28]), we introduced several strategies to choose examinations and time-slots using ordering heuristics within the framework of squeaky wheel optimization. This work is an extension of the adaptive heuristic orderings technique proposed by [29] where the approach promotes difficult examinations to be scheduled first at each of iteration using a *heuristic modifier*. In an other study, [30] implemented an adaptive approach to examination timetabling by hybridizing the low level graph heuristics based on a learning mechanism and modifying the solutions by high-level heuristic indirectly.

With most of the approaches taken within the overall constructive family of methods, it is often the case that a single heuristic is used during the initial ordering phase. In considering the difficulty of an examination, it is useful to take into account other factors that effect the ordering of examinations. Considering many factors at one time represents the real world situation. The difficulty of scheduling an examination can be approximated more realiably if several heuristics lend support to the final ordering of examinations. The current constructive study by [31] combined graph colouring heuristics with weights within liner approach as to measure the difficulty of a vertex of weighted graph. The study used the vertex-selection heuristics to represent the difficulty of a vertex and it is continually updated throughout the timetabling process. Studies by [32] and [21] also have deployed this strategy by considering more than one heuristic at one time and it has been shown that it has an effect on the ordering of the examinations. Based on the '*difficulty factor*', [32] used graph colouring heuristics i.e. the combination of largest enrolment and largest degree as ordering strategy to assign examinations to time slots. Several variation of relative weight of each criterion was considered in order to produce several different feasible timetables. [21] combined two graph colouring heuristics under the framework of fuzzy methodology in order to deal with uncertainty in ordering the examination based on

its difficulties. The solution quality appeared to be superior compared to only using one heuristic. Encouraged by these studies, we extend this work by combining heuristics with a heuristic modifier and adapting different weights to each heuristic to analyse its effectiveness. The effect of weights associated with ordering using different heuristics on the quality of the examination schedules is investigated. It is worthy of note that by using information from these heuristics and a heuristic modifier, improvements can be seen in the obtained solution. This approach has been shown to produce high quality solutions that are comparable to other approaches in the literature ([5]).

A review on the adaptive ordering heuristics is explained in Section 2. Section 3 provided the implementation, the instances we focus on and the analysis of the results. Finally the conclusion is provided in Section 4 with some future direction.

## 2. A Linear Combination of Heuristics Orderings

An adaptive approach to examination timetabling based on priorities was proposed by [29]. This approach was extended by [28] who introduced additional strategies to improve the solution quality by including methods to choose the ordering of examinations and their assignment to time-slots. The method is based on the idea of squeaky wheel optimisation initiated by [33]. Squeaky wheel optimisation is a greedy approach and works by iteratively cycling around three procedures: *Constructor*, *Analyzer* and *Prioritizer*.

In examination timetabling problems, squeaky wheel optimisation gives priority to a *'difficult'* examination so that it is chosen earlier in the next iteration. In relation to the examination timetabling problem, the process is as follows:

- *Constructor*. First, the constructor generates an initial solution for a set of unscheduled examinations based on the initial ordering (which can be generated by a chosen graph colouring heuristic). The unscheduled examinations are individually assigned to the best time-slot i.e. whichever generates least penalty. During the assignment, there is a possibility that some of the examinations cannot be assigned to a time-slot due to the existence of conflicts with other examinations. In this case, such examinations remain unscheduled.

- *Analyzer*. Once the constructor has completed the assignment, each examination is analysed to check whether the examination had a problem during the assignment. A strategy is used to increase the priority of the problematic examination so that it will be given more priority during the next iteration. A certain value is topped up to the difficulty value of the unscheduled examination in order to indicate that this unscheduled examination is more difficult to handle than other examinations. This difficulty value will therefore increase at the end of each iteration if an examination remains unscheduled during the assignment.

- *Prioritizer*. Increasing the difficulty value in certain ways may change the ordering of examinations. At this stage, the updated difficulty value will be ranked in decreasing order and the most difficult examination will be chosen to schedule first in the next iteration. The process continues until some stopping criteria is met and finally the best solution found is returned.

In order to modify the difficulty value of an examination over time, we used the idea of a *heuristic modifier* introduced by [29]. The formula for examination difficulty is presented in equation 1. The difficulty of examination $i$ at iteration $t$ is a discrete variable that is an estimation of the difficulty of scheduling the examination after completing the iteration while the heuristic of examination $i$ is a chosen graph colouring heuristic value that estimates the difficulty. The $heurmod_i(t)$ for examinations $i$ at iteration $t$ is a *heuristic modifier* value. It is an integer value that will increase in certain ways using a *modify* function during the iteration if examination $i$ is unscheduled (equation 1).

$$difficulty_i(t) = heuristic_i + heurmod_i(t) \tag{1}$$

where,

$$heurmod_i(t+1) = \begin{cases} modify(heurmod_i(t)) & , \text{if unscheduled} \\ heurmod_i(t) & , \text{otherwise} \end{cases}$$

A linear combination of heuristic orderings is a combination of a number of normalized graph colouring heuristics and normalized difficulty measures from the *heuristic modifier* introduced by [29]. It is a flexible approach because different weights can be assigned to different parameters used in the combination. Information from the chosen

heuristics and heuristic modifier are used to identify new orderings of examination to be scheduled. The new ordering of an examination based on a linear combination of heuristic orderings is represented by the following equation:

$$difficulty\_score_i(t) = \sum_{j=1}^{n} w_j \times heuristicN_{ij} + w_{HM} \times heurmod_i(t) \tag{2}$$

where,

$$heuristicN_{ij} = \frac{heuristic_{ij}}{maxheuristic_j} \tag{3}$$

$$heurmodN_i(t) = \frac{heurmod_i(t)}{choosemax(heurmod(t))} \tag{4}$$

$$\sum_{j=1}^{n} w_j + w_{HM} = 1 \tag{5}$$

,

The $difficulty\_score_i(t)$ is used as a difficulty measure for examination $i$ at iteration $t$ based on the information evaluated. In this study, the zero-one normalisation method is used to obtain the normalised value for each $heuristicN_{ij}$ and $heurmodN_i(t)$ to ensure a simple generalisation characteristic of the problem data. The $heuristicN_{ij}$ in equation 3 is the normalised graph colouring heuristic $j$ for examination $i$ while $heurmodN_i(t)$ in equation 4 is the normalised heuristic modifier for examination $i$ at iteration $t$. The $maxheuristic_j$ is the maximum identified value of heuristic $j$ while the $choosemax$ function is provided to give an alternative to the heuristic modifier to change dynamically or statically. Given in equation 5 is the total weight for heuristic $j$, $w_j$ and heuristic modifier, $w_{HM}$ is equal to 1.

### 2.1. Graph Colouring Heuristics

Two types of graph colouring heuristics were used in this suite of experiments. In order to compare their contribution to solution quality, a series of experiments has been carried out, firstly, combining both heuristics and subsequently using each single heuristic separately. The purpose here was to compare the performance of single and multiple heuristics and to identify the most effective combination of heuristics with the heuristic modifier. Note that, although not investigated here, more graph colouring heuristics can be used within this approach.

- *Largest Degree (LD).* The examinations are ordered decreasingly based on the number of conflicts with other examinations where the examination with the largest conflict will be scheduled first. On the other hand, the $heuristic_{ij}$ holds the number of conflicting examinations of heuristic $j$ (largest degree) for examination $i$. The $heuristicN_{ij}$ is the normalised value of $heuristic_{ij}$ with $maxheuristic$ i.e. the largest number of conflicting examination. This heuristic is classified as a static heuristic because of the heuristic value for each examination remains unchanged throughout the iteration.

- *Saturation Degree (SD).* The ordering of examinations is based on the number of remaining time-slot where the examination with the least time-slot will be scheduled first. The remaining time-slot of an examination is the number of time-slots that has no confliction with an examination. This heuristic is classified as a dynamic heuristic. The number of remaining time-slots of unscheduled examination will keep changing as the conflicting examinations are assigned to time-slots. Also, the ordering of unscheduled examination may change because of the current assignment. The saturation degree value in this study is initialised with 1 and will keep increasing until the maximum number of time-slots if the examination cannot be scheduled during the iteration. This value is vice-versa from the original saturation degree as we want to keep the difficulty of an examination to increase. As for capacitated problem i.e. the problem with room capacity requirement, the saturation degree value also considered the availability of rooms for the remaining time-slot. Let say, all the remaining time-slots for an examination are feasible. However, three of the remaining time-slots are infeasible due to no rooms available for those time-slots. In this circumstances, the number of remaining time-slots will be reduced. The next examination to be scheduled is determined by the $difficulty\_score$ of the remaining unscheduled examination, where the largest value of the $difficulty\_score$ will be scheduled first. In this approach, $heuristic_{ij}$ holds the number of remaining time-slot of examination $i$. The $maxheuristic$ of saturation degree is the total number of time-slot given for the dataset.

## 2.2. Heuristic Modifiers (HM)

In [28], a different *modify* function for heuristic modifiers was used to change the order of examinations based on its difficulty value. The difficulties were updated and increased with four strategies i.e. custom, additive, multiplicative and exponential. In this study, we employed only the additive and exponential strategy since in this approach we adapted the normalization strategy in order to generalize the ordering of *difficulty_score*.

- *Additive (AD)*. The additive modifier works by incrementing the difficulty of an examination by 1 during each iteration where the examination cannot be scheduled. Incrementing the difficulty using this strategy has a modest effect because it requires more time to increase in order to emphasize the degree of difficulty of an examination.

$$modify(heurmod_i(t)) = heurmod_i(t - 1) + 1, heurmod_i(0) = 0 \qquad (6)$$

- *Exponential (EX)*. This exponential modifier will upgrade the difficulty of an examination exponentially, if the examination is difficult to schedule. The examination order will change significantly due to the large increment in the difficulty value.

$$modify(heurmod_i(t)) = c \times heurmod_i(t - 1), heurmod_i(0) = 1, where\ c = 2 \qquad (7)$$

Once the heuristic modifier and the difficulty of an examination have been updated, the difficulty value will be normalized statically or dynamically based on the *choosemax* function. After all the heuristic values and the heuristic modifier have been updated with the chosen weights, all the values will be summed up to get the *difficulty_score*. The examinations will then be ordered decreasingly based on the *difficulty_score* before an assignment is made. The pseudocode is described in Algorithm 1.

---

**Algorithm 1** Construction of a timetable based on linear combination of heuristic orderings.

---

**for** $t$ = 0 to MAXIter **do**
  **for** $i$ = 0 to MAXExam **do**
    **for** $j$ = $i$ to MAXExam **do**
      Calculate the normalise value of choosen heuristics:   $choosemax(heurmodN_i(t))$,   $HeuristicN_{LD,i}$, $HeuristicN_{SD,i}$ with weight value
      Calculate the $difficulty\_score_i(t)$ for each exam according to the chosen heuristics
    **end for**
    Sort($difficulty\_score(t)$) in decreasing order
    **if** $i$ can be scheduled **then**
      Schedule $i$ in the time-slot with the least penalty
      In the case of the availability of multiple time-slots with the same penalty, choose one randomly
    **else**
      Increase and $modify(heurmod_i(t))$
    **end if**
    **if** Saturation_degree **then**
      Update the Saturation_degree
    **end if**
  **end for**
  Evaluate solution, store if it is the best found so far
**end for**

---

## 2.3. The choosemax Function

The normalized value of the heuristic modifier is determined by the *choosemax* function that gives significant modification to the $heurmodN_i(t)$.

- *Static (S)*. The *heurmod$_i$(t)* is normalized with the total number of iterations used in the algorithm. The larger the *heurmod$_i$(t)*, the more significant the value of *heurmodN$_i$(t)*.

- *Dynamic (D)*. The *heurmod$_i$(t)* is normalized with the maximum number of *heurmod* of all examinations that change during the iteration. This value will keep changing until the end of the iteration.

## 2.4. The weight assigment

Each of the heuristic and the heuristic modifiers has its weight. In this approach, we assigned the weight values from 0 to 1 with a 0.1 increment for each variable with the total of all weights are equal to 1 (equation (6)). We have tested the combination of these values for each of the variables in order to see the performance of the heuristics and the heuristic modifier. It is important to know which heuristic is performing well and to see the importance of the heuristic modifier in this combination so that the higher weight value will be given to the appropriate parameters.

## 2.5. Shuffling the Ordering of Examinations

In this study, we employ the shuffling strategy introduced in [29]. We adapted the *top-window* (TW) strategy to choose examination in the order. Examinations are ordered based on the *difficulty_score* and from a fixed size of top-window, an examination is chosen randomly. The motivation for this strategy is to give more possibility to the most appropriate examination to be chosen first. The appropriate examination to be chosen might appear in a certain size of grouped examinations that has been ordered based on the *difficulty_score*. In this study, we used the top-window size from two to nine as suggested in [28]. In another strategy, since there is also possibility that examinations have the same value of *difficulty_score* we introduced *random* preference (REQ) in order that an examination can be chosen differently from the original ordering if equal *difficulty_score* occurs for several sequences of examinations.

## 2.6. Time-slot Choice

Once an examination has been chosen to be scheduled, it is assigned to the most appropriate time-slot. The assignment is made referring to the least penalty cost considering all the available time-slots. The time-slot with the least penalty cost will be chosen. Since there is a possibility that some time-slots generate the same least penalty, we have incorporated a random element in making this choice and introduced variation of assignments in the timetable.

Tables 2.6 and 2.6 show an example of how the ordering is done using various combinations of heuristics. Table 2.6 illustrate the single heuristic. In this example, the total number of time-slots is 10 and the maximum number of iterations is 100. The weight value for the single heuristic is set to 1 and the other heuristics are set as 0. In column 1, the single ordering for LD is based on equation 3 where the *maxheuristic* for largest degree is equal to 19. The *difficulty_score* for e4 = 19/19 = 1.0, e1 = 17/19 = 0.89 and so on.

Table 1: Examples of ordering by combinations of single heuristics (LD=largest degree; SD=saturation degree; HM=heuristic modifier; diff_score = difficulty_score).

| Unordered list | | Ordering by single LD | | Ordering by single SD | | Ordering by single HM | |
|---|---|---|---|---|---|---|---|
| exams | LD | exams | diff_score | exams | diff_score | exams | diff_score |
| e1 | 17 | e4 | 1.0 | e2 | 0.1 | e2 | 0.50 |
| e2 | 14 | e1 | 0.89 | e4 | 0.2 | e4 | 0.48 |
| e3 | 16 | e10 | 0.84 | e6 | 0.3 | e6 | 0.30 |
| e4 | 19 | e3 | 0.84 | e10 | 0.4 | e8 | 0.28 |
| e5 | 9 | e2 | 0.74 | e3 | 0.5 | e10 | 0.25 |
| e6 | 11 | e6 | 0.58 | e1 | 0.6 | e3 | 0.20 |
| e7 | 8 | e5 | 0.47 | e9 | 0.6 | e7 | 0.15 |
| e8 | 8 | e7 | 0.42 | e5 | 0.6 | e1 | 0.10 |
| e9 | 8 | e9 | 0.42 | e7 | 0.6 | e9 | 0.08 |
| e10 | 16 | e8 | 0.42 | e8 | 0.6 | e5 | 0.05 |

The example for SD is shown in column 2. The single ordering for SD is dynamic. After each assignment of a time-slot, the new examination ordering is obtained. Initially, the saturation degree value is set to 1. By using equation 1, e2 has been chosen as the first examination to be assigned to the time-slot and the *difficulty_score* for e2 = 1/10 = 0.1 . Once e2 has been assigned to a time-slot, the saturation degree value will be updated by considering the conflict with other examinations in a previous assignment. Let only e4 have conflicts with e2, then the saturation degree of e4 will be increased by one and the *difficulty_score* for e4 = 2/10 = 0.2. The ordering by single heuristic modifier (HM) in column 3 in Table 2.6 is based on the number of time an examination cannot be scheduled during the iteration. By considering equation 1, the *difficulty_score* for e2 = 50 (number of time e2 cannot be scheduled)/100 = 0.50, e4 = 48/100 = 0.48 and so on.

Table 2: Examples of ordering by combinations of multiple heuristics (LD=largest degree; SD=saturation degree; HM=heuristic modifier; diff_score=difficulty_score).

| Ordering by LDSD | | Ordering by LDHM | | Ordering by LDSDHM | |
|---|---|---|---|---|---|
| exams | diff_score | exams | diff_score | exams | diff_score |
| e4 | 0.28 | e4 | 0.70 | e4 | 0.488 |
| e1 | 0.34 | e1 | 0.57 | e1 | 0.451 |
| e3 | 0.41 | e10 | 0.49 | e10 | 0.408 |
| e10 | 0.49 | e3 | 0.41 | e3 | 0.408 |
| e2 | 0.55 | e2 | 0.37 | e2 | 0.419 |
| e6 | 0.60 | e6 | 0.32 | e6 | 0.415 |
| e5 | 0.57 | e5 | 0.24 | e5 | 0.375 |
| e7 | 0.56 | e8 | 0.23 | e9 | 0.396 |
| e9 | 0.56 | e9 | 0.20 | e8 | 0.388 |
| e8 | 0.56 | e7 | 0.19 | e7 | 0.388 |

Table 2.6 illustrates the example combination of more than one heuristic. The ordering by LDSD is a dynamic ordering. Consider the weight for LD = 0.2 and the weight for SD = 0.8. By using equation 2, the *difficulty_score* for this combination for e4 = (0.2)(19/19) + (0.8)(0.1) = 0.28, for e1 = (0.2)(17/19) + (0.8)(0.2) = 0.34 and so on based on the combination of information from largest degree and saturation degree. Let us consider the weight for LD = 0.2 and the weight for HM = 0.8 for ordering LDHM. The *difficulty_score* for e4 = (0.2)(19/19) + (0.8)(0.62) = 0.70, e1 = (0.2)(17/19) + (0.8)(0.48) = 0.57 etc. In the next combination of heuristics, let consider the weight for LD = 0.2, the weight for SD = 0.4 and the weight HM = 0.4 for ordering LDSDHM. The *difficulty_score* for e4 = (0.2)(19/19) + (0.4)(0.1) + (0.4)(0.62) = 0.488, e1 = (0.2)(17/19) + (0.4)(0.2) + (0.4)(0.48) = 0.451 etc.

## 3. Experiments

In the experiment described, two benchmark problems are tested. Due to the stochastic nature of the proposed approaches, 50 different timetables were constructed for each of datasets from the Toronto and the second international timetabling competition (ITC2007) benchmark. Various combinations of heuristics and heuristic modifier were considered in order to determine and compare the performance of the proposed approaches. Different weights were also assigned to each heuristic and heuristic modifier with the total weight equal to one for each of combination. The stopping conditioned for this approach is set as 2000 iterations for the Toronto while, the experiment for ITC2007 is based on the running time given in the competition. The best combination of heuristics and heuristic modifier is identified based on the best solution obtained. The best penalty value obtained from 50 runs is highlighted in bold for each problem instance.

### 3.1. Experimental Data

The Toronto benchmark datasets used in this study were introduced by [34] and it is widely used as test bed in the examination timetabling community. There are 13 sets of problems from various universities around the world

with different problem dimensions and characteristics. These dataset is publicly available and can be accessed at ftp://ftp.mie.utoronto.ca/pub/carter/testprob/. Since there is a problem relating to the circulation of datasets under the same name, [5] has introduced notations to differentiate various version of the dataset. In this study, we adapt the introduced notation to specify the dataset. We used version I of the datasets and the characteristic of the dataset are specified in Table 3.1.

Table 3: The characteristic of the Toronto benchmark ([34]).

| Problem | No. of time-slots | No. of exams | No. of students | Conflict density (%) |
|---------|-------------------|--------------|-----------------|----------------------|
| car92   | 32                | 543          | 18 419          | 0.14                 |
| car91   | 35                | 682          | 16 925          | 0.13                 |
| ears83 I| 24                | 190          | 1 125           | 0.27                 |
| hec92 I | 18                | 81           | 2 823           | 0.42                 |
| kfu93   | 20                | 461          | 5 349           | 0.06                 |
| lse91   | 18                | 381          | 2 726           | 0.06                 |
| pur93 I | 42                | 2419         | 30 032          | 0.03                 |
| rye92   | 23                | 486          | 11 483          | 0.08                 |
| sta83 I | 13                | 139          | 611             | 0.14                 |
| tre92   | 23                | 261          | 4 360           | 0.18                 |
| uta92 I | 35                | 622          | 21 266          | 0.13                 |
| ute92   | 10                | 184          | 2 750           | 0.08                 |
| yor83 I | 21                | 181          | 941             | 0.29                 |

The objective of the Toronto benchmark problem is to create a feasible timetable so that no students sitting two examinations at one time and at the same time students are assigned as apart as possible in order to give more student spread in the timetable. We used the proximity cost function introduced by [34] in conjunction with the introduced datasets. This cost function is used to measure the quality of the obtained timetable and to describe the average penalty of students spread in the examination schedule.

The ITC2007 is broken into three problems i.e. the examination timetabling, post enrolment based course timetabling and curriculum based course timetabling that reflect the real problem within educational timetabling with the focus to Higher Education ([35]). The aim of the competition is to build better understanding between researchers and practitioners of real-life problem by allowing new implementation to the introduced problem. This study is focused on the examination timetabling problem of the introduced datasets. The mathematical formulation of the problem can be refer in [36]. Differ from the Toronto datasets, the ITC2007 is a capacitated problem that required room assignment for each of examination. Other than the conflict free requirement as hard constraint, the ITC2007 is also required to fulfill the time-slot and room hard constraint requirements. For more details on the constraints and the mathematical formulation see [35] and [36]. The characteristics for each dataset of ITC2007 can be view in Table 3.1.

### 3.2. Experimental Result
### 3.2.1. Toronto

The experimental results for different combinations of graph colouring heuristics are provided in Table 3.2.1. The results show the best penalty value obtained from 50 runs for different combinations of heuristics and is highlighted in bold for each problem instance. The comparison shows that the combination of LDSDHM performed the best with ten out of thirteen datasets and one tie with SDHM, while SDHM has obtained two best results for two datasets. This circumstance shows that by considering information from more than one parameter simultaneously, the new difficulty measure can be obtained and at the same time a new ordering of examinations can be generated. As can be seen, the single SD performed well in comparison to the single LD and this is may be because of the dynamic nature of this heuristic. The single HM also performed well and has obtained the best six out of thirteen datasets if compared to the other single heuristics.

Table 4: The characteristic of the ITC2007 benchmark ([35], [36])(HC = hard constraint).

| Problem | No. of time-slots | No. of exams | No. of students | No. of rooms | No. of time-slots (HC) | No. of rooms (HC) | Conflict density |
|---|---|---|---|---|---|---|---|
| Exam_1 | 54 | 607 | 7 891 | 7 | 12 | 0 | 5.05 |
| Exam_2 | 40 | 870 | 12 743 | 49 | 12 | 2 | 1.17 |
| Exam_3 | 36 | 934 | 16 439 | 48 | 170 | 15 | 2.62 |
| Exam_4 | 21 | 273 | 5 045 | 1 | 40 | 0 | 15.0 |
| Exam_5 | 42 | 1018 | 9 253 | 3 | 27 | 0 | 0.87 |
| Exam_6 | 16 | 242 | 7 909 | 8 | 23 | 0 | 6.16 |
| Exam_7 | 80 | 1096 | 14 676 | 15 | 28 | 0 | 1.93 |
| Exam_8 | 80 | 598 | 7 718 | 8 | 20 | 1 | 4.55 |
| Exam_9 | 25 | 169 | 655 | 3 | 10 | 0 | 7.84 |
| Exam_10 | 32 | 214 | 1 577 | 48 | 58 | 0 | 4.97 |
| Exam_11 | 26 | 934 | 16 439 | 40 | 170 | 15 | 2.62 |
| Exam_12 | 12 | 78 | 1 653 | 50 | 9 | 7 | 18.45 |

Table 5: Comparison of single and combination of heuristics (LD=largest degree; SD=saturation degree; HM=heuristic modifier).

| Problem | LD | SD | HM | LDSD | LDHM | SDHM | LDSDHM | Best |
|---|---|---|---|---|---|---|---|---|
| car91 | 5.32 | 5.26 | 5.43 | 5.22 | 5.25 | 5.18 | **5.12** | 5.12 |
| t(s) | 468 | 387 | 888 | 357 | 642 | 389 | 368 | |
| car92 | 4.61 | 4.58 | 4.63 | 4.55 | 4.49 | 4.42 | **4.41** | 4.41 |
| t(s) | 259 | 22 | 487 | 208 | 365 | 222 | 215 | |
| ears83 | 38.41 | 39.83 | 39.52 | 38.62 | 38.47 | 39.06 | **36.91** | 36.91 |
| t(s) | 27 | 24 | 25 | 24 | 29 | 24 | 25 | |
| hec92 | 11.7 | 11.68 | 11.72 | 11.52 | 11.52 | 11.42 | **11.31** | 11.31 |
| t(s) | 4 | 4 | 3 | 4 | 4 | 5 | 4 | |
| kfu93 | 15.24 | 14.97 | 15.53 | 14.97 | 15.08 | **14.75** | 14.93 | 14.75 |
| t(s) | 106 | 234 | 422 | 127 | 128 | 236 | 137 | |
| lse91 | 12.23 | 11.98 | 11.79 | 11.55 | 11.64 | 11.51 | **11.41** | 11.41 |
| t(s) | 72 | 97 | 260 | 75 | 77 | 98 | 80 | |
| pur93 | 5.93 | 6.05 | 6.42 | 5.93 | 5.95 | 5.92 | **5.87** | 5.87 |
| t(s) | 229 | 361 | 1586 | 290 | 821 | 877 | 920 | |
| rye92 | 10.25 | 9.89 | 9.76 | 9.95 | 9.65 | **9.61** | 9.63 | 9.61 |
| t(s) | 131 | 202 | 521 | 172 | 136 | 194 | 156 | |
| sta83 | 158.63 | 158.08 | 157.75 | 157.84 | 157.97 | 157.77 | **157.52** | 157.52 |
| t(s) | 10 | 14 | 11 | 9 | 10 | 14 | 11 | |
| tre92 | 3.61 | 3.67 | 3.67 | 3.6 | 3.59 | **3.54** | **3.54** | 3.54 |
| t(s) | 358 | 292 | 850 | 283 | 416 | 295 | 302 | |
| uta92 | 27.14 | 26.92 | 26.79 | 26.79 | 26.55 | 26.27 | **26.25** | 26.25 |
| t(s) | 12 | 20 | 24 | 13 | 13 | 19 | 14 | |
| ute92 | 9.25 | 8.94 | 8.85 | 8.94 | 8.76 | 8.81 | **8.76** | 8.76 |
| t(s) | 45 | 43 | 55 | 42 | 53 | 43 | 41 | |
| yor83 | 41.88 | 40.96 | 41.48 | 40.73 | 41.1 | 40.08 | **39.67** | 39.67 |
| t(s) | 22 | 25 | 26 | 22 | 26 | 24 | 22 | |

Table 3.2.1 shows the combination of weights and algorithmic approaches for the best results obtained from the experiments. This shows that most of the best results were obtained using the dynamic heuristic modifier. The value of dynamic heuristic modifier is updated by finding highest value of heuristic modifier at each time the assignment process is done. Considering the shuffling strategy, best results are obtained on ten out of thirteen datasets using the top-window strategy, while random preference worked effectively better with three out of thirteen datasets. From the table, weight for HM is highest for six of the datasets, four of the dataset obtained the best result with high weight value for SD and the other three datasets performed well with LD as highest value of weight.

Table 6: Different combination of weights and algorithmic approaches (LD=largest degree; SD=saturation degree; HM=heuristic modifier; St=static; Dy=dynamic; REQ=random preference; TW=top-window; AD=additive; EX=exponential; w=weight).

| Problem | Best | wLD | wSD | wHM |
|---------|------|-----|-----|-----|
| car92 I | 4.41 {LDSDHM, Dy, TW(4), AD} | 0.2 | 0.3 | 0.5 |
| car91 I | 5.12 {LDSDHM, Dy, REQ, EX} | 0.1 | 0.2 | 0.7 |
| ears83 I | 36.91 {LDSDHM, St, TW(4), AD} | 0.3 | 0.1 | 0.6 |
| hec92 I | 11.31 {LDSDHM, Dy, TW(3), AD} | 0.2 | 0.5 | 0.3 |
| kfu93 | 14.75 {SDHM, Dy, TW(4), EX} | 0.0 | 0.1 | 0.9 |
| lse91 | 11.41 {LDSDHM, Dy, REQ, EX} | 0.1 | 0.5 | 0.4 |
| pur93 I | 5.87 {LDSDHM, St, TW(4), AD} | 0.2 | 0.6 | 0.2 |
| rye92 | 9.61 {SDHM, Dy, REQ, EX} | 0.0 | 0.1 | 0.9 |
| sta83 I | 157.52 {LDSDHM, Dy, REQ, EX} | 0.5 | 0.4 | 0.1 |
| tre92 | 3.54 {LDSDHM, Dy, REQ, EX} | 0.2 | 0.2 | 0.6 |
| uta92 I | 26.25 {LDSDHM, Dy, REQ, EX} | 0.8 | 0.1 | 0.1 |
| ute92 | 8.76 {LDSDHM, Dy, REQ, EX} | 0.8 | 0.1 | 0.1 |
| yor83 | 39.67 {LDSDHM, Dy, REQ, EX} | 0.1 | 0.8 | 0.1 |

Table 3.2.1(a), 3.2.1(b) and 3.2.1(c) reports the comparison of results for the thirteen problem instances of Toronto benchmark datasets for three different groups of approaches i.e. constructive, hyper-heuristic and other improvement approaches, respectively. The comparison are made with other approaches that has been published in journal article and the best result of each group is highlighted in bold font. Comparison of constructive heuristic approaches shows that the linear combination approach obtained one best result for sta83 I. Meanwhile, some other results of the linear combination approach are very close to the best of constructive approaches such as ute92 and yor83 I. The comparison to other hyper-heuristic approaches shows that the linear combination approach obtained five best results out of thirteen problem instances on hec92 I, rye92, sta83 I, ute92 and yor83 I. Note that the hyper-heuristic approaches may incorporated a two phase algorithm i.e. construction and improvement. Meanwhile, the linear combination approach is purely a constructive algorithm that construct the examination timetables using heuristic ordering. On the other hand, the comparison with other improvement approaches shows that most of the results from the linear combination approach are quite far from the best results of improvement approaches. But still some are better than the other improvement approaches such as car91, car92, rye92, sta83 I, tre92 and uta92 I.

In order to see the different of solution quality of the proposed approach when using different top-window size and different group of weight combination, a two analysis of variance is performed. From the statistical analysis, $F_{(31,58156)} = 18.750$ and $\rho(0.000) < 0.05$ and it shows that there is significant differences to solution quality when different top-window size and group of weight combination were employed. Table 3.2.1 illustrates the different between the tested top-window size from the algorithm. It shows that in most cases different top-window size can performed significantly different to each other. However, the top-window size 2 is not significantly different from size 3 while the size 3 is not significantly different from size 2 and 4.

The weight combinations are divided into four different groups based on the heuristic contribution. From the initial test of the weight combination, it shows that there is not much different when using different type of weight combination since there are just little differences between the weight combinations. For instance, the weight combination of (0.1, 0.1, 0.8) is not much different with weight combination of (0.2, 0.1, 0.7) in terms of the performance of solution quality since the weight combination is almost the same. In this case, the weight combinations are divided

Table 7: Comparison for different approaches (a) constructive heuristics, (b) hyper-heuristics and (c) other improvement approaches. (LC = our approach)

| Problem | [34] | [29] | [21] | [31] | LC |
|---------|------|------|------|------|------|
| car91 | 7.1 | **4.97** | 5.29 | 5.03 | 5.12 |
| car92 | 6.2 | 4.32 | 4.54 | **4.22** | 4.41 |
| ears83 I | 36.4 | 36.16 | 37.02 | **36.06** | 36.91 |
| hec92 I | **10.8** | 11.61 | 11.78 | 11.71 | 11.31 |
| kfu93 | **14** | 15.02 | 15.8 | 16.02 | 14.75 |
| lse91 | **10.5** | 10.96 | 12.09 | 11.15 | 11.41 |
| pur93 I | **3.9** | - | - | - | 5.87 |
| rye92 | **7.3** | - | 10.38 | 9.42 | 9.61 |
| sta83 I | 161.5 | 161.9 | 160.4 | 158.86 | **157.52** |
| tre92 | 9.6 | 8.38 | 8.67 | **8.37** | 8.76 |
| uta92 I | 3.5 | **3.36** | 3.57 | 3.37 | 3.54 |
| ute92 | **25.8** | 27.41 | 28.07 | 27.99 | 26.25 |
| yor83 I | 41.7 | 40.77 | 39.8 | **39.53** | 39.67 |

(a)

| Problem | [19] | [37] | [38] | [30] | LC |
|---------|------|------|------|------|------|
| car91 | 5.36 | **4.97** | 5.16 | 5.17 | 5.12 |
| car92 | 4.53 | 4.28 | **4.16** | 4.32 | 4.41 |
| ears83 I | 37.92 | 36.86 | 35.86 | **35.7** | 36.91 |
| hec92 I | 12.25 | 11.85 | 11.94 | 11.93 | **11.31** |
| kfu93 | 15.2 | **14.62** | 14.79 | 15.3 | 14.75 |
| lse91 | 11.33 | **11.14** | 11.15 | 11.45 | 11.41 |
| pur93 I | - | **4.73** | - | - | 5.87 |
| rye92 | - | 9.65 | - | - | **9.61** |
| sta83 I | 158.19 | 158.33 | 159 | 159.05 | **157.52** |
| tre92 | 8.92 | **8.48** | 8.6 | 8.68 | 8.76 |
| uta92 I | 3.88 | 3.4 | 3.42 | **3.3** | 3.54 |
| ute92 | 28.01 | 28.88 | 28.3 | 28 | **26.25** |
| yor83 I | 41.37 | 40.74 | 40.24 | 40.79 | **39.67** |

(b)

| Problem | [15] | [17] | [39] | [40] | LC |
|---------|------|------|------|------|------|
| car91 | 5.2 | 6.6 | **4.6** | 4.8 | 5.12 |
| car92 | 4.4 | 6 | **3.9** | 4.1 | 4.41 |
| ears83 I | 34.9 | **29.3** | 32.8 | 34.92 | 36.91 |
| hec92 I | 10.3 | **9.2** | 10 | 10.73 | 11.31 |
| kfu93 | 13.5 | 13.8 | **13** | 13 | 14.75 |
| lse91 | 10.2 | **9.6** | 10 | 10.01 | 11.41 |
| pur93 I | - | **3.7** | - | 4.73 | 5.87 |
| rye92 | 8.7 | **6.8** | - | 9.65 | 9.61 |
| sta83 I | 159.2 | 158.2 | **156.9** | 158.26 | 157.52 |
| tre92 | 8.4 | 9.4 | 7.9 | **7.88** | 8.76 |
| uta92 I | 3.6 | 3.5 | 3.2 | **3.2** | 3.54 |
| ute92 | 26 | **24.4** | 24.8 | 26.11 | 26.25 |
| yor83 I | 36.2 | 36.2 | **34.9** | 36.22 | 39.67 |

(c)

Table 8: The effect of different size of top-window for LDSDHM.

| Size | Effect | Size |
|------|--------|------|
| 2 | $\neq$ | (4, 5, 6, 7, 8 and 9 ($p = 0.000$)) |
| 2 | $\simeq$ | (3 ($p = 0.110$)) |
| 3 | $\neq$ | (5, 6, 7, 8 and 9 ($p = 0.000$)) |
| 3 | $\simeq$ | (2 ($p = 0.110$)) and (4 ($p = 0.089$)) |
| 4 | $\neq$ | (2, 5, 6, 7, 8 and 9 ($p = 0.000$)) |
| 4 | $\simeq$ | (3 ($p = 0.089$)) |
| 5 | $\neq$ | (2, 3, 7, 8 and 9 ($p = 0.000$)), (4 ($p = 0.018$)) and (6 ($p = 0.025$)) |
| 6 | $\neq$ | (2, 3, 4, 8 and 9 ($p = 0.000$)), (5 ($p = 0.025$)) and (7 ($p = 0.001$)) |
| 7 | $\neq$ | (2, 3, 4, 5, 8 and 9 ($p = 0.000$)) and (6 ($p = 0.001$)) |
| 8 | $\neq$ | (2, 3, 4, 5, 6 and 7 ($p = 0.000$)) and (9 ($p = 0.001$)) |
| 9 | $\neq$ | (2, 3, 4, 5, 6 and 7 ($p = 0.000$)) and (8 ($p = 0.001$)) |

into four different groups i.e. highLD, highSD, highHM and balance. The details of weight combinations of each group is shown in Table 3.2.1. The result from the two way analysis of variance in Table 3.2.1 shows that the solution quality of each group is statistically different. In this circumstances, the solution quality that is tested with the weight combination from any different group of heuristic are statistically has differences.

Table 9: The grouping of different weight combinations for LDSDHM.

| Group | Weight Combination |
|-------|--------------------|
| High LD | (0.8, 0.1, 0.1), (0.7, 0.2, 0.1), (0.7, 0.1, 0.2) |
| High SD | (0.1, 0.8, 0.1), (0.1, 0.7, 0.2), (0.2, 0.7, 0.1) |
| High HM | (0.1, 0.1, 0.8), (0.1, 0.2, 0.7), (0.2, 0.1, 0.7) |
| Balance | (0.3, 0.3, 0.4), (0.4, 0.3, 0.3), (0.3, 0.4, 0.3) |

Table 10: The effect of different group of weight combination for LDSDHM without considering top-window size.

| Group | Effect | Group |
|-------|--------|-------|
| High LD | $\neq$ | (High SD , High HM and Balance ($p = 0.000$)) |
| High SD | $\neq$ | (High LD , High HM and Balance ($p = 0.000$)) |
| High HM | $\neq$ | (High LD , High SD and Balance ($p = 0.000$)) |
| Balance | $\neq$ | (High LD , High SD and High HM ($p = 0.000$) |

Figure 3.2.1(a) and 3.2.1(b) illustrates the best performance of LDSDHM for car92 I and tre92 considering all combinations of weight. The figures show a pattern in the performance of best solution quality obtained for each of the combination. By looking at the median value, when the weight value of HM is high enough then the solution quality value will rapidly drop. On the other hand, whenever the weight value of HM is gradually decreased, then the solution quality is also decreased gradually. Most of the peaks are obtained from the lowest weight value of HM while most of the slump are obtained from the highest weight value of HM. This shows that the existence of the heuristic modifier in this linear combination of heuristics has an effect to the solution quality. Furthermore, by using the information from the other two heuristics it has increased the effectiveness of the new ordering. The results indicate that the combinations are most effective when the weight of HM is very high while the other heuristics may vary in certain ways.
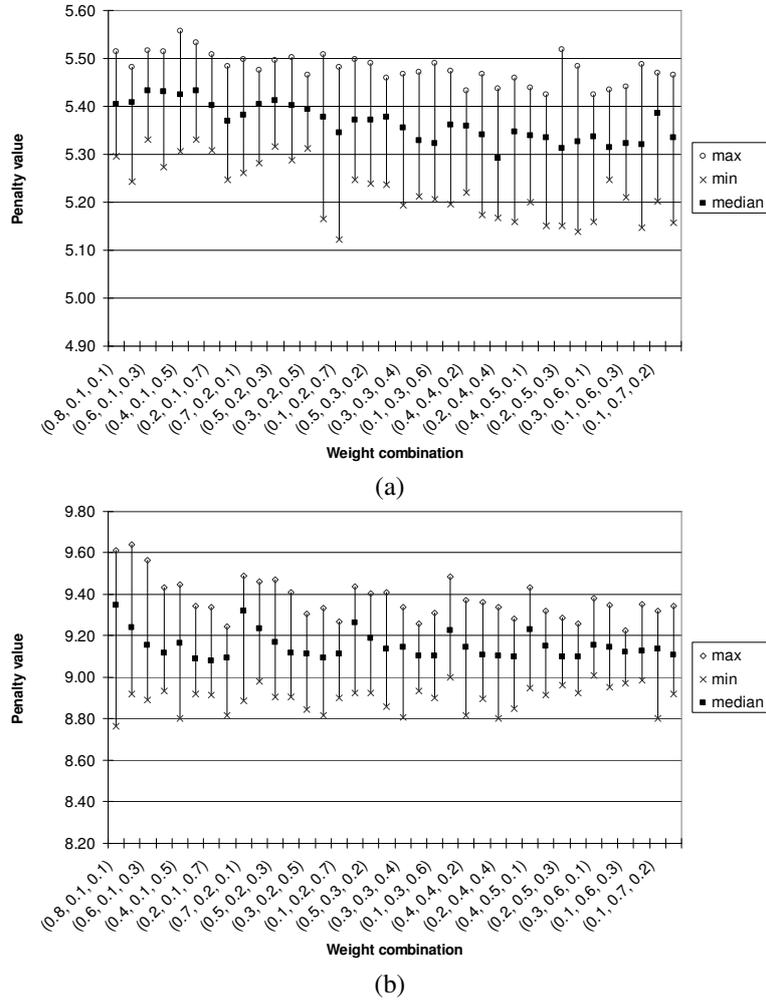
(a)



(b)

Figure 1: Best solution quality for each of weight combination of LDSDHM for (a) car91 I and (b) tre92.
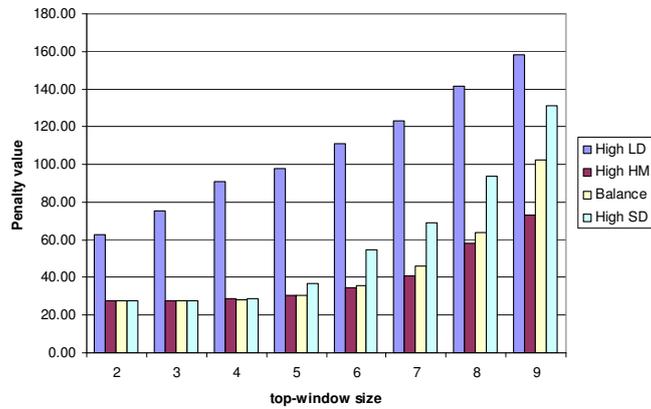


Figure 2: Average performance of each top-window size and different group of weight combination for LDSDHM.

Figure 2 illustrates the average performance of each top-window size and and different group of weight combination for LDSDHM. As can be seen, the group of highLD contributes higher penalty value at each top-window size while the highSD, highHM and balance are almost the same in terms of penalty value during smaller size of top-window. However, the average performance for highSD, highHM and balance started to differ when the top-window size is 6 and above. In this circumstances, using a smaller chunk of top-window size with a good choice of weight combinations may lead to a better quality solution.

### 3.2.2. ITC2007

The experiment on the twelve problem instances of ITC2007 was tested with several combination of weights with only top-window size 3 and 5. The heuristic modifier is increased using additive and exponential with dynamic modification of heuristic modifier value. Table 11 illustrates the best penalty value obtained from 50 runs and each solution is provided with the information of weight combination and algorithmic approach. As shown in Table 11, the weight combination of each heuristic for different instances are showing the different pattern. As can be seen, about half of the problem instances obtained good quality solution when the weight of the SD is high. Meanwhile, the weight LD and HM are varied in certain ways. Since the ITC2007 datasets is tested with only certain parameter setting not as Toronto benchmark datasets and due to time limitation, these datasets might not showing the exact pattern of the whole weight behavior. Moreover, the ITC2007 dataset is a capacitated timetabling problem and it is different from the Toronto benchmark datasets in terms of various hard and soft constraints requirement.

Table 11: Different combination of weights and algorithmic approaches for ITC2007 datasets (LD = largest degree; SD = saturation degree; HM = heuristic modifier; Dy = dynamic; TW = top-window; AD = additive; EX = exponential; w = weight).

| Problem | Best | wLD | wSD | wHM |
|---------|------|-----|-----|-----|
| Exam_1 | 11060 {LDSDHM Dy TW(5) EX} | 0.5 | 0.3 | 0.2 |
| Exam_2 | 3133 {LDSDHM Dy TW(3) AD} | 0.4 | 0.1 | 0.5 |
| Exam_3 | 19098 {LDSDHM Dy TW(3) EX} | 0.1 | 0.7 | 0.2 |
| Exam_4 | 21309 {LDSDHM Dy TW(3) AD} | 0.3 | 0.6 | 0.1 |
| Exam_5 | 7975 {LDSDHM Dy TW(3) EX} | 0.3 | 0.5 | 0.2 |
| Exam_6 | 28330 {LDSDHM Dy TW(5) EX} | 0.1 | 0.8 | 0.1 |
| Exam_7 | 15912 {LDSDHM Dy TW(5) AD} | 0.1 | 0.8 | 0.1 |
| Exam_8 | 20066 {LDSDHM Dy TW(3) EX} | 0.7 | 0.1 | 0.2 |
| Exam_9 | 2165 {LDSDHM Dy TW(3) AD} | 0.4 | 0.2 | 0.4 |
| Exam_10 | 16516 {LDSDHM Dy TW(3) AD} | 0.1 | 0.3 | 0.6 |
| Exam_11 | 45873 {LDSDHM Dy TW(3) AD} | 0.1 | 0.6 | 0.3 |
| Exam_12 | 7465 {LDSDHM Dy TW(3) AD} | 0.7 | 0.2 | 0.1 |

Table **??** shows the results of different strategies of weight changes for the ITC2007 datasets. The best quality solutions are highlighted in bold font. Clearly, it shows that the dynamic weight strategy performed the best among the three strategies with 8 problem instances, while the linear weight and reinforcement learning strategies performed the best with 3 and 1 problems instances, respectively. Even though the dynamic weight strategy performed well, nevertheless, this strategy is unable to produce feasible solution for the Exam_4. This is also proved by [41] whereby this problem instance is difficult to solve. As the ITC2007 instances involved lots of constraint requirements, the dynamic weight changes might be an advantage as weight combination is identified based on the cost of each successive assignment. However, as mentioned before this strategy may spent lots of time in calculating the examinations assignment cost. Note that the experiments on the ITC2007 follows the running time requirement as ruled out in the competition.

As so far, most of the ITC2007 approaches have concentrated on the multiple phases of solution that construct and improve the solution quality in sequence. The linear combination approach in this chapter is presented as constructive approach that iteratively construct the examination timetable. As can be seen in Table 12, it shows the comparison of the best penalty values of the linear combination approach compared with other approaches from the competition and post-competition. As can be seen , in any ways, the results of the linear combination approach cannot beat the best

results obtained so far and are quite far from them. However, the linear combination approach is able to produced feasible solution for all problem instances and are better than some of the approaches from the competition and post-competition. The proposed approach is able to produce better results compared to [42] for Exam_10, [43] for Exam_2, Exam_4 and Exam_6, [44] for Exam_1, Exam_4, Exam_6, Exam_7 and Exam_10, [45] for Exam_4 and Exam_6 and also [46] for Exam_4, Exam_6 and Exam_8. On the other hand, some of the approaches do not have solutions for some of the problem instances for example [47] for Exam_10 and Exam_12, [43] for Exam_11, [48] for Exam_3, Exam_4, Exam_8, Exam_11 and Exam_12 while [45], [46] and [40] do not have solution for Exam_9, Exam_10, Exam_11 and Exam_12.

Table 12: Comparison of LC with different approaches of ITC2007 datasets. (LC = linear combination approach)

| Problem | [49] | [47] | [43] | [48] | [44] | [42] | [41] | [45] | [46] | [50] | [40] | LC |
|---------|------|------|------|------|------|------|------|------|------|------|------|------|
| Exam_1 | 4370 | 5905 | 8006 | 6670 | 12035 | 4370 | 4633 | 8559 | 6235 | 4775 | **4368** | 11060 |
| Exam_2 | 400 | 1008 | 3470 | 623 | 3074 | 385 | 405 | 830 | 2974 | **385** | 390 | 3133 |
| Exam_3 | 10049 | 13862 | 18622 | - | 15917 | 9378 | 9064 | 11576 | 15832 | **8996** | 9830 | 19098 |
| Exam_4 | 18141 | 18674 | 22559 | - | 23582 | **15368** | 15663 | 21901 | 35106 | 16204 | 24822 | 20830 |
| Exam_5 | 2988 | 4139 | 4714 | 3847 | 6860 | 2988 | 3042 | 3969 | 4873 | **2929** | 3022 | 7975 |
| Exam_6 | 26950 | 27640 | 29155 | 27815 | 32250 | 26365 | 25880 | 28340 | 31756 | **25740** | 25995 | 28330 |
| Exam_7 | 4213 | 6683 | 10473 | 5420 | 17666 | 4138 | **4037** | 8167 | 11562 | 4087 | 4067 | 15912 |
| Exam_8 | 7861 | 10521 | 14317 | - | 16184 | 7516 | **7461** | 12658 | 20994 | 7777 | 7519 | 20066 |
| Exam_9 | 1047 | 1159 | 1737 | 1288 | 2055 | 1014 | 1071 | - | - | **964** | - | 2165 |
| Exam_10 | 16682 | - | 15085 | 14778 | 17724 | 14555 | 14374 | - | - | **13203** | - | 16516 |
| Exam_11 | 34129 | 43888 | - | - | 40535 | 31425 | 29180 | - | - | **28704** | - | 45124 |
| Exam_12 | 5535 | - | 5264 | - | 6310 | 5357 | 5693 | - | - | **5197** | - | 7465 |

## 4. Conclusion

In this paper, a linear combination of heuristics with a heuristic modifier under the framework of adaptive strategies is proposed for solving examination timetabling problems. Two graph colouring heuristics with a heuristic modifier are adapted with different weights for each parameter. Each parameter is normalised in order to simply generalised the implemented problem data. A *difficulty_score* is used to determine the ordering of the examinations and the most difficult examination with the highest *difficulty_score* will be scheduled first based on two strategies. This approach is tested with single and multiple heuristics with and without a heuristic modifier on Toronto and ITC2007 datasets. The results show that by combining multiple heuristics with a heuristic modifier, a good solution quality can be obtained. Furthermore, the results from the combination of LDSDHM are comparable to the results of other constructive approaches published in the literature within the Toronto benchmark problems. Meanwhile, the results on the highly constraint ITC2007 problems are feasible and some are comparable to the previous approaches. In this study, the combination of weight values that are invoked to the heuristics and heuristic modifier could significantly change the examination ordering based on the difficulty_score value. It is found that by changing the weight values of the heuristic and heuristic modifier the heuristic modifier could obtain good approximate solutions. It is also identified that the best top-window size to use for this approach is six and below as the higher value of top-window size could cause the significant change in the examination ordering. It is concluded that this approach is simple and effective, hence it has potential for practical use.

## References

[1] D. Schindl, Some new hereditary classes where graph coloring remains np-hard, Discrete Mathematics 295 (1-3) (2005) 197–202.
[2] B. McCollum, A perspective on bridging the gap between research and practice in university timetabling, in: E.K.Burke, H.Rudova (Eds.), Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science., Vol. 3867, Springer, 2007, pp. 3–23.

[3] M. W. Carter, A survey of practical applications of examination timetabling algorithms, Operational Research 34 (2) (1986) 193–202.

[4] M. W. Carter, G. Laporte, Recent developments in practical examination timetabling, in: Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Springer-Verlag, London, UK, 1996, pp. 3–21.

[5] R. Qu, E. K. Burke, B. Mccollum, L. T. Merlot, S. Y. Lee, A survey of search methodologies and automated system development for examination timetabling, Journal of Scheduling 12 (1) (2009) 55–89.

[6] E. K. Burke, D. Elliman, P. H. Ford, R. F. Weare, Examination timetabling in british universities: A survey, in: Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Springer-Verlag, London, UK, 1996, pp. 76–90.

[7] E. K. Burke, J. Kingston, D. de Werra, Handbook of Graph Theory, Chapman Hall/CRC Press, 2004, Ch. Applications to timetabling, pp. 445–474.

[8] E. K. Burke, J. P. Newall, A multistage evolutionary algorithm for the timetable problem, IEEE Trans. Evolutionary Computation 3 (1) (1999) 63–74.

[9] E. K. Burke, J. P. Newall, R. F. Weare, A memetic algorithm for university exam timetabling, in: E. K. Burke, P. Ross (Eds.), Lecture notes in computer science: Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Vol. 1153, Springer-Verlag, London, UK, 1996, pp. 241–250.

[10] E. Ersoy, E. Özcan, A. Sima Uyar, Memetic algorithms and hyperhill-climbers, in: G. Baptiste, P.and Kendall, A. M. Kordon, F. Sourd (Eds.), Lecture notes in computer science: Multidisciplinary international conference on scheduling: theory and applications: selected papers from the 3rd international conference, 2007, pp. 159–166.

[11] G. M. White, B. S. Xie, S. Zonjic, Using tabu search with longer-term memory and relaxation to create examination timetables, European Journal of Operational Research 153 (1) (2004) 80–91.

[12] Z. N. Azimi, Hybrid heuristics for examination timetabling problem, Applied Mathematics and Computation 163(2) (2005) 705–733.

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

[14] J. M. Thompson, K. A. Dowsland, A robust simulated annealing based examination timetabling system, Comput. Oper. Res. 25 (7-8) (1998) 637–648.

[15] S. Abdullah, S. Ahmadi, E. K. Burke, M. Dror, Investigating ahuja-orlin's large neighbourhood search approach for examination timetabling, OR Spectrum 29 (2) (2007) 351–372.

[16] R. Qu, E. K. Burke, Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems, in: Proceedings of the 6th Metaheuristic International Conference 2005, Vienna, Austria, 2005.

[17] M. Caramia, P. Dell'Olmo, G. Italiano, Novel local search based approaches to university examination timetabling., INFORMS Journal of Computing 20 (2008) 86–99.

[18] S. Casey, J. Thompson, Grasping the examination scheduling problem, in: Lecture notes in computer science: Practice and theory of automated timetabling IV: selected papers from the 4th international conference, Vol. 2740, Berlin: Springer, 2003, pp. 234–244.

[19] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper-heuristic for educational timetabling problems, European Journal of Operational Research 176 (2007) 177–192.

[20] E. K. Burke, S. Petrovic, R. Qu, Case based heuristic selection for timetabling problems, Journal of Scheduling 9 (2) (2006) 115–132.

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

[22] P. Boizumault, Y. Delon, L. Peridy, Constraint logic programming for examination timetabling, Journal of Logic Programming 26 (2) (1996) 217–233.

[23] S. K. Abdul Rahim, A. Bargiela, R. Qu, Granular modelling of exam to slot allocation, in: Paper presented at the 23rd European Conference on Modelling and Simulation (ECMS), Madrid, Spain, 2009, pp. 861–866.

[24] S. Petrovic, E. Burke, University timetabling, in: Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chapter 45, Chapman Hall/CRC Press, 2004.

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

[26] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, S. Schulenburg, Handbook of Meta-Heuristics, Kluwer, 2003, Ch. Hyper-Heuristics: An Emerging Direction in Modern Search Technology, pp. 457–474.

[27] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, Handbook of Metaheuristics, Vol. 146, Springer, 2010, Ch. A classification of hyper-heuristic approaches, pp. 449–468.

[28] S. Abdul Rahman, A. Bargiela, E. K. Burke, B. McCollum, E. Özcan, Construction of examination timetables based on ordering heuristics, in: Proceedings of the 24th International Symposium on Computer and Information Sciences, 2009, pp. 727–732.

[29] E. K. Burke, J. P. Newall, Solving examination timetabling problems through adaptation of heuristic orderings, Annals of Operations Research 129 (2004) 107–134.

[30] R. Qu, E. Burke, B. McCollum, Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems, European Journal of Operational Research 198 (2) (2009) 392–404.

[31] E. K. Burke, N. Pham, R. Qu, J. Yellen, Linear combinations of heuristics for examination timetabling, Annals of Operations Researchdoi:10.1007/s10479-011-0854-y.

[32] D. Johnson, Timetabling university examinations, Journal of the Operational Research Society 41 (1) (1990) 39–47.

[33] D. E. Joslin, D. P. Clements, "squeaky wheel" optimization, Journal of Artificial Intelligence Research 10 (1999) 353–37.

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

[35] B. McCollum, P. McMullan, E. K. Burke, A. J. Parkes, R. Qu, A new model for automated examination timetabling, Annals of OR 2 (2008) 2–3.

[36] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, E. K. Burke, Setting the research agenda in automated timetabling: The second international timetabling competition, INFORMS JOURNAL ON COMPUTING 22 (1) (2010) 120–130.

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

[38] R. Qu, E. K. Burke, Hybridisations within a graph based hyper-heuristic framework for university timetabling problems, Journal of Operational Research Society 60 (2009) 1273–1285.

[39] E. K. Burke, A. J. Eckersley, B. McCollum, S. Petrovic, R. Qu, Hybrid variable neighbourhood approaches to university exam timetabling, European Journal of Operational Research 206 (1) (2010) 46–53.

[40] H. Turabieh, S. Abdullah, An integrated hybrid approach to the examination timetabling problem., Omega 39 (2011) 598–607.

[41] B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, S. Abdullah, An extended great deluge approach to the examination timetabling problem, in: The 4th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA09), Dublin, 2009.

[42] T. Müller, Itc2007 solver description: a hybrid approach, Annals OR 172 (1) (2009) 429–446.

[43] M. Atsuta, K. Nonobe, T. Ibaraki, Itc2007 track 1: An approach using general csp solver, www.cs.qub.ac.uk/itc2007.

[44] N. Pillay, A developmental approach to the examination timetabling problem., in: Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008., 2008, www.cs.qub.ac.uk/itc2007.

[45] N. Pillay, Evolving hyper-heuristics for a highly constrained examination timetabling problem., in: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland., 2010, pp. 336–346.

[46] E. Burke, R. Qu, A. Soghier, Adaptive selection of heuristics for improving constructed exam timetables., in: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), 10-13 August 2010, Belfast, Northern Ireland, 2010, pp. 136–152.

[47] C. Gogos, P. Alefragis, E. Housos, A multi-staged algorithmic process for the solution of the examination timetabling problem, in: Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008., 2008.

[48] G. De Smet, Itc2007 - examination track, in: Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008., 2008.

[49] T. Müller, Itc 2007: Solver description., in: Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, 19-22, August 2008., 2008.

[50] C. Gogos, P. Alefragis, E. Housos, An improved multi-staged algorithmic process for the solution of the examination timetabling problem., Annals of Operation Research 3 (2010) 1–3. doi:10.1007/s10479-010-0712-3.