Adaptive Linear Combination of Heuristic Orderings in Constructing Examination Timetable

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

[a] *Universiti Utara Malaysia, School of Quantitative Sciences, 06010 Sintok, Malaysia*
[b] *University of Nottingham, School of Computer Science, Jubilee Campus, Nottingham NG8 1BB, UK*
[c] *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 an adaptive 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 an adaptive 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 obtained promising results. We conclude that this adaptive linear combination of heuristics is a highly effective method and simple to implement.

Keywords: examination timetabling, 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 ([5]). 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]).

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 ([5]), there are many different such constraints that have been highlighted by different academic institutions in Britain. Examples of real-world application within examination timetabling problem is in [35].

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 ([2]). 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 [6].

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 ([2]). Many other approaches upon meta-heuristic technique and hybridisation have shown great success such as evolutionary algorithm ([7]), tabu search ([8]), ant algorithm ([9]) and simulated annealing ([10]). 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 ([11]), variable neighbourhood search ([27]) and iterated local search ([12]). The reliant on parameter setting of these approaches has lead to the introduction of other approaches such as hyper-heuristic ([13]), case-based reasoning ([14]), fuzzy approaches ([15]) and granular information processing ([16]) within the timetabling arena. A review on major approaches in examination timetabling can be found in [3] and [4].

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 ([2], [6]). 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 ([17]).

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

([18]), we introduced several strategies to choose examinations and time-slots using ordering heuristics within the framework of squeaky wheel optimisation. This work is an extension of the adaptive heuristic orderings technique proposed by [19] where the approach promotes difficult examinations to be scheduled first at each of iteration using a *heuristic modifier*. In an other study, [20] implemented an adaptive approach to examination timetabling by hybridising 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 family of constructive 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 affect 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 reliably if several heuristics lend support to the final ordering of examinations. Consequently, the current constructive study by [21] 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 [22] and [15] have also 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*', [22] used graph colouring heuristics i.e. the combination of largest enrolment and largest degree as ordering strategy for assigning examinations to time slots. Several variations of relative weight of each criterion were considered in order to produce a number of different feasible timetables. Further, [15] combined two graph colouring heuristics within the framework of fuzzy methodology in order to deal with uncertainty in ordering the examination based on its difficulties. Three graph colouring heuristics were used, i.e. largest degree, largest enrolment and saturation degree with three combinations of two heuristics. The study indicated that the solution quality was superior compared with using only a single 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 aim is to obtain new difficulty values that are extracted from the combination of graph colouring heuristics with a heuristic modifier using a linear approach. Different weights are assigned to each parameter and 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 tested on two datasets i.e. Toronto and ITC2007 (The Second International Timetabling Competition) benchmark datasets and have shown to produce high quality solutions that are comparable to other approaches in the literature ([4]).

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. An Adaptive Linear Combination of Heuristics Orderings

An adaptive approach to examination timetabling based on priorities was proposed by [19]. This approach was extended by [18] 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 [23]. Squeaky wheel optimisation is a greedy approach and works by iteratively cycling around three procedures: *Constructor*, *Analyzer* and *Prioritizer*. In relation to the examination timetabling problem, the procedures are 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 the 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 there was a problem related with the assignment i.e. whether there is conflict with other examinations during the assignment. A strategy is used to increase the priority of the problematic examination so that it will be given a higher priority

in the next iteration. A certain value is added 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 by adding a certain value to a heuristic may change the ordering of examinations. At this stage, the updated difficulty value will be ranked in a decreasing order and the most difficult examination will be chosen to be scheduled first in the next iteration. The process continues until some stopping criterion is met and finally the best solution found is returned.

This approach constructs the solution considering the hard constraint of the tested problem and the quality of the constructed solution is measured based on the soft constraint violation. The type of hard and soft constraints and the evaluation of the quality of the constructed timetable differ across different problem instances. For further detail on the hard and soft constraint evaluation on the tested benchmark datasets see [4] and [25].

In order to modify the difficulty value of an examination over time, the idea of a *heuristic modifier* introduced by [19] is used. 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. $heurmod_i(t)$ for examination i at iteration t is a *heuristic modifier* value. At each iteration, $heurmod_i(t)$ is increased by a modify function whenever examination i cannot be scheduled (illustrated in equation 1). This approach can be considered as an online learning algorithm where the feedback from the search process while solving the problem is used to construct the next solution during the iteration.

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

where,

$$heurmod_i(t{+}1) = \begin{cases} modify(heurmod_i(t)) & \text{, if examination } i \text{ cannot be scheduled} \\ heurmod_i(t) & \text{, otherwise} \end{cases}$$

An adaptive linear combination of heuristic orderings in this study is a combination of a number of normalised graph colouring heuristics with normalised difficulty measures from the *heuristic modifier*. 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 an adaptive linear combination of heuristic orderings is represented by the following equation:

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

where,

$$heuristicN_{ij} = \frac{heuristic_{ij}}{maxheuristic_j} \quad (3)$$

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

$$\sum_{j=1}^{n} w_j + w_{HM} = 1 \quad (5)$$

The $difficulty\_score_i(t)$ is used as a difficulty measure for examination i at iteration t based on the information evaluated. In the present study, a zero-one normalisation method is used to obtain the normalised value between 0 and 1 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, using each single heuristic separately, and subsequently combining both heuristics with and without a heuristic modifier. The purpose is to compare the performance of single and multiple heuristics and to identify the most effective combination of heuristics with the heuristic modifier. It should be noted that, although not investigated here, more graph colouring heuristics can be used within this approach.

- *Largest Degree (LD).* The ordering of examinations is based on the number of conflicting examinations where the examination with the largest conflict will be scheduled first. The $heuristic_{ij}$ holds the number of conflicting examinations of heuristic j (e.g. 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. In solving the un-capacitated problem, the examination to be scheduled is checked whether they are conflicted with other examinations in assigning the time-slot and room. If the assignment cannot be made, then the $heurmod_i(t)$ is increased twice. Subsequently, it also increased the difficulty score of examination i.

- *Saturation Degree (SD).* The ordering of examinations is based on the number of remaining time-slots where the examination with the smallest number of available time-slots is scheduled first. The number of remaining time-slots of unscheduled examinations will keep changing as the conflicting examinations are assigned to time-slot. 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. The ordering of unscheduled examinations may change due to the current successive assignment. Since the saturation degree value of an examination decreases from time to time, it requires an adjustment. In this study, the complement of an examination is used where the saturation degree of an examination is $(max\_number\_of\_time\_slot - saturation\_degree\_of\_an\_examination)$. The saturation degree value is initialised with 0 and keeps increasing until the maximum number of time-slots is reached if the examination cannot be scheduled during the iteration. The complement of saturation degree is used to increase the difficulty of an examination by adding

it to the heuristic modifier. As for the capacitated problem (i.e. the problem with room capacity requirement), its saturation degree value also considers the availability of rooms for the remaining time-slot. For example, the number of remaining time-slots of an examination that can be used to schedule is seven. Assuming that three of the remaining time-slots are considered invalid due to the unavailability of rooms. In these circumstances, the number of remaining time-slots of an examination that can be scheduled is reduced to four considering room availability at the same time. Using this heuristic, the priority of choosing an examination is given to the higher value of difficulty. The next examination to be scheduled is determined by the $difficulty\_score$ of the remaining unscheduled examination, where the largest value of the $difficulty\_score$ is scheduled first. In this approach, $heuristic_{ij}$ holds the complement of 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 our previous study, [18] used various *modify* function for heuristic modifiers to change the order of examinations based on their difficulty value. The difficulty values were updated and increased with four strategies: custom, additive, multiplicative and exponential. The examination ordering was based on only one graph colouring heuristic during the timetabling. In this study, the same modify function i.e. additive and exponential are employed as in [18] and the linear approach adapts the normalisation strategy in order to generalise the ordering of $difficulty\_score$ by combining a number of graph colouring heuristic with a heuristic modifier.

- *Additive (AD).* The modifier is increased by one at each iteration, if an examination cannot be scheduled. This strategy has a modest effect on the difficulty of a given examination. If the difference between the heuristic value of a given examination i and its predecessor in the priority list is large, then it will take longer in using this approach to reorder the given examination i, emphasizing that this examination i is difficult to schedule:

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

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

$$modify(heurmod_i(t)) = c \times heurmod_i(t-1), heurmod_i(0) = 1, \quad (7)$$

where $c = 2$.

Once the heuristic modifier and the difficulty of an examination have been updated, the difficulty value of the heuristic modifier is normalised 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 are summed up to obtain $difficulty\_score$. The examinations are then ordered decreasingly based on $difficulty\_score$ before an assignment is made. The pseudocode of the implemented approach for the Toronto benchmark datasets is described in Algorithm 1. It is worthy to note that the Toronto benchmark datasets are un-capacitated problem.

The initial ordering of examinations at the beginning of the timetabling process is set based on the largest degree graph colouring heuristic. The saturation degree value of each examination i is set to be equal to 0 at the beginning of each iteration if it is chosen as the heuristic of the algorithm. This saturation degree value will keep increasing up to the maximum number of time-slots. Once the timetabling process begins, the normalise value for each of the chosen heuristics for each examination j is calculated using equation 2, 3 and 4. Next, the new difficulty score for each examination j is obtained and they are sorted in decreasing order. The most difficult examination (examination i) with the highest difficulty score is chosen first for time-slot assignment and is checked for the hard constraint violation. If examination i can be scheduled, then it is scheduled with the least penalty time-slot. In the case of more than one same least penalty time-slot available, then the best time-slot is selected randomly from the list. In the case of examination i being violated, then it is left unassigned and at this stage the heuristic modifier of examination i is increased based on the identified type of modify function of the heuristic modifier. For the saturation degree heuristic, the saturation degree value for each examination i is updated after each successive examination assignment. The process of the examination assignment using adaptive linear combination of heuristic orderings continues

Algorithm 1 Constructing an examination timetable based on adaptive linear combination of heuristic orderings for the Toronto benchmark datasets

Choose heuristic(s) and do the initial ordering based on LD
Assign weight for each of the chosen heuristics
for $t = 1$ to number of iterations **do**
  if Saturation_degree **then**
    Set the saturation degree for each examination as 0
  end if
  for $i = 1$ to number of examinations **do**
    for $j = i$ to number of examinations **do**
      Calculate the normalise value of chosen heuristics: $choosemax(heurmodN_j(t))$, $HeuristicN_{LD,j}$, $HeuristicN_{SD,j}$ (refer equation 2, 3 and 4) with weight value
      Calculate the $difficulty\_score_j(t)$ for each examination according to the chosen heuristics using equation 1
    end for
    Sort($difficulty\_score(t)$) in a decreasing order
    Choose examination i that has the highest difficulty score to be scheduled
    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 heuristic modifier of i (refer equation 6 and 7)
    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

until all examinations i has been assigned to time-slot. At the end of the process, the solution quality of the constructed timetable is evaluated and the best solution quality is stored.

2.3. The choosemax *Function*

The normalised 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 normalised 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 normalised with the current maximum number of *heurmod* of all examinations that change during the iteration. This value continues to change until the end of the iteration.

2.4. The weight assignment

Since this approach required weight assignment for each parameter, this study needs a strategy to assign the weight value. Each of the heuristics and the heuristic modifiers is assigned with different weight values. Using this approach, the weight values are assigned to each heuristic and heuristic modifier with the value from 0 to 1 with a 0.1 increment for each variable. The total of all weight values is equal to 1 (equation 6). The combination of these weight values is tested for each of the variables in order to assess the performance of the heuristics and the heuristic modifier when different weight values are incorporated. It is important to know which heuristic is performing well by obtaining good quality solutions and to note the importance of the heuristic modifier in this combination, so that the higher weight value is given to the appropriate parameters.

2.5. Shuffling the Ordering of Examinations

The present study employed the shuffling strategy used in our previous study [18] in order to shuffle the examinations in the ordering, where the *top-window* (TW) strategy is used to choose examinations. These are ordered based on the *difficulty_score* and from a fixed size of top-window, an examination is chosen randomly. The insight of this strategy is to give more possibility to an examination to be chosen from a group of difficult examinations. An appropriate examination to be chosen might appear in a certain size

of grouped examinations that has been ordered based on the *difficulty_score*. The initial test has shown that the incorporation of the shuffling strategy could assist in finding a better examinations ordering. This study uses the top-window size from two to nine, as suggested by [18]. Since there is also a possibility that examinations have the same value of difficulty score, another strategy introduces a *random* preference (REQ) in order to choose different examinations when several sequences of examinations have equal scores.

2.6. Time-slot Choice

Once an examination is chosen, it is assigned to the most appropriate time-slot. The assignment is made to ensure the smallest penalty cost from among all the available time-slot assignments. Previous studies [**?** 19, 31] have indicated that the first time-slot that generates the least penalty is chosen for an assignment. Since there is a possibility that some time-slots generate the same least penalty, a random element is incorporated in making this choice, introducing a variation of assignments in the timetable. In such a situation, there is a possibility of an examination being assigned to a different time-slot during another iteration, even though the order of examinations in the current iteration is the same as in the previous iteration.

2.7. Illustration of the Implementation

Tables 1 and 2 show an example of how the ordering is achieved using various combinations of heuristics after a certain number of iterations. In this example, the total number of time-slots is 10. Table 1 illustrates the ordering using a single heuristic. Since we want to use only one heuristic for the ordering, then the weight value for the single heuristic that is chosen is set to 1.0 and the weight for other heuristics are set as 0. Referring to column 2 of an unordered list in Table 1, the values are assumed as largest degree values. In that case, the *maxheuristic* for largest degree is equal to 19. The calculation of the *difficulty_score* value for the single ordering of LD in column 5 is based on equation 1 where the difficulty score for e4 = (1.0)19/19 = 1.0, e1 = (1.0)17/19 = 0.89 and so on.

The example for SD is shown in column 3 of Table 1. The single ordering for SD is dynamic. After each assignment of a time-slot, the new examination ordering is obtained. Initially, as implemented by [18], the saturation degree value is set to 0 for all examinations. Assumed that e2 is chosen as the first examination to be assigned to a time-slot. Once e2 is assigned to a time-slot, the saturation degree value for the unscheduled examinations is updated by

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	HM	exams	diff_score	exams	diff_score	exams	diff_score
e1	17	4	e4	1.00	e2	-	e2	1.00
e2	14	20	e1	0.89	e4	0.1	e4	0.75
e3	16	10	e10	0.84	e6	0.1	e6	0.70
e4	19	15	e3	0.84	e10	0.1	e8	0.60
e5	9	0	e2	0.74	e3	0.0	e10	0.60
e6	11	14	e6	0.58	e1	0.0	e3	0.50
e7	8	7	e5	0.47	e9	0.0	e7	0.35
e8	8	12	e7	0.42	e5	0.0	e1	0.20
e9	8	0	e9	0.42	e7	0.0	e9	0.00
e10	16	12	e8	0.42	e8	0.0	e5	0.00

considering the conflict with other examinations in a previous assignment. Assumed that e4, e6 and e10 have conflicts with e2, then the saturation degree of these examinations are increased by one and the *difficulty_score* (using equation 1) for e4 = (1.0)1/10 = 0.1, e6 = (1.0)1/10 = 0.1 and e10 = (1.0)1/10 = 0.1, while the difficulty value for the rest of the examinations are zero due to no conflict with e2. The calculation of the difficulty value for each unassigned examination continues until no more examinations are to be assigned to a time-slot. The ordering by single heuristic modifier (HM) in column 1 in Table 1 is based on the number of times an examination cannot be scheduled during the previous iterations. It is assumed that the figures in column 1 are the number of times these examinations cannot be assigned into a timetable during the previous iterations. By considering equation 1, the *difficulty_score* for ordering by single HM for e2 = (1.0)20/20= 1.0, e4 = (1.0)15/20 = 0.75, e6 = (1.0)14/20 = 0.70 and so on.

Table 2 illustrates the example of combinations of more than one heuristic. It is assumed that the total number of time-slot to be assigned is 10. The ordering by LDSD is a dynamic ordering. Considering the weight for LD, $w_{LD} = 0.2$ and the weight for SD, $w_{SD} = 0.8$. Furthermore, it is assumed

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
e2	-	e4	0.800	e2	-
e4	0.280	e6	0.676	e4	0.500
e1	0.258	e10	0.648	e10	0.408
e3	0.168	e3	0.568	e6	0.396
e10	0.168	e8	0.564	e3	0.368
e6	0.116	e7	0.364	e8	0.324
e5	0.095	e1	0.279	e1	0.299
e7	0.084	e2	0.147	e7	0.224
e9	0.084	e5	0.095	e5	0.095
e8	0.084	e9	0.084	e9	0.084

that e2 is the first examination to be chosen for assignment and it has been assigned to a time-slot and assuming also that e2 has conflict only with e4 and e1. In this case, the saturation degree values for e4 and e1 are increased by 1. By using equation 1 and considering the largest degree value from Table 1, the *difficulty_score* for this combination for e4 = (0.2)(19/19) + (0.8)(1/10) = 0.280, for e1 = (0.2)(17/19) + (0.8)(1/10) = 0.258, e3 = (0.2)(16/19) + (0.8)(0/10) = 0.168 and so on, where these calculations are based on the combination of information from the largest degree and saturation degree heuristics. Let us consider the weight for LD, wLD = 0.2 and the weight for HM, wHM = 0.8 for ordering the examinations using combination of LDHM. Considering the largest degree and HM values from Table 1, the *difficulty_score* (equation 1) for e4 = (0.2)(19/19) + (0.8)(15/20) = 0.800, e6 = (0.2)(11/19) + (0.8)(14/20) = 0.676 , e10 = (0.2)(16/19) + (0.8)(12/20) = 0.648 etc. In the next combination of heuristics, let us consider the weight for LD, wLD = 0.2, the weight for SD, wSD = 0.4 and the weight HM, wH M = 0.4 for ordering the examinations with combination of LDSDHM. It is assumed that e2 is the first examination to be chosen for assignment at certain iteration and it has been assigned to a time-slot and has conflict only with e4 and e1. Considering the information from Table 2, the *difficulty_score* for e = (0.2)(19/19) + (0.4)(1/10) + (0.4)(15/20) = 0.500, e10 = (0.2)(16/19)

+ (0.4)(0/10) + (0.4)((12/20) = 0.408, e6 = (0.2)(11/19) + (0.4)(0/10) + (0.4)(14/20) = 0.396 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 dataset from the Toronto and ITC2007 benchmark with each combination of heuristic(s) and heuristic modifier. Various combinations of heuristic(s) and heuristic modifier are considered in order to determine and compare the performance of the proposed approaches. Different weights are also assigned to each heuristic and heuristic modifier with the total weight equal to one for each combination. The stopping condition 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 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 [24] and it is widely used as test bed in the examination timetabling community with different problem dimensions and characteristics. These datasets 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, [4] introduced notations to differentiate various version of the datasets. In this study, the notation introduced is used to specify the datasets and version I is used as a test bed to the proposed approaches.

The objective of the Toronto benchmark problem is to create a feasible timetable so that no student is required to sit two examinations at any one time. To achieve a high quality timetable, the soft constraints need to be satisfied as much as possible. Thus, during the timetable construction, it is required that student's examinations are assigned as far apart as possible in order to give a wider student spread in the timetable. The proximity cost function introduced in [24] in conjunction with the introduced datasets was used in order to measure the quality of the obtained timetable and to describe the average penalty of students distributed in the examination schedule.

The ITC2007 datasets are used for the evaluation of the approach proposed in this paper, where the focuses are on the examination timetabling track. It differs from the Toronto datasets in that ITC2007 is a capacitated

problem that requires room assignment for each examination. Moreover, time-slot-related constraints and room-related constraints are also considered as the hard constraints to be adhered to. In order to obtain a good quality timetable, several new soft constraints are also taken into account to fulfill the real world requirements; there are seven soft constraints to be satisfied simultaneously with their contribution to the quality of the obtained timetable. For more details on the constraints and the mathematical formulation of the ITC2007 datasets see [25].

3.2. Experimental Result

3.2.1. Toronto

The results of the experiments for different combinations of graph colouring heuristics are provided in Table 3. The results show the best penalty value obtained from 50 runs for different combinations of heuristics. The comparison shows that the combination of LDSDHM performed the best with ten out of thirteen datasets and one tie with SDHM, while SDHM obtained 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. It can be seen that the single SD performed well in comparison with the single LD and this may be because of the dynamic nature of this heuristic. The single HM also performed well and obtained the best results for six out of thirteen datasets when compared with the other single heuristics.

Table 4 illustrates the combination of weights and algorithmic approaches for the best results obtained from the experiments. It shows that most of the best results are obtained using the dynamic heuristic modifier. The value of the dynamic heuristic modifier is updated by finding the highest value of heuristic modifier each time the assignment process is completed. Taking the shuffling strategy into account, best results are obtained for ten out of thirteen datasets using the top-window strategy, while random preference works more effectively with three out of thirteen datasets. As observed in the table, the weight for HM is the highest for six of the datasets, while 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.

The best approaches so far within Toronto benchmark datasets are described by [12], [27] and [28]. The study by [12] investigated a multi phase local search based algorithms that starts with a greedy scheduler to create a

Table 3: Comparison of single and combination of heuristics for the Toronto benchmark datasets (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 I	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 I	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 I	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 I	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	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	
uta92	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	
ute92 I	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	
yor83 I	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	

feasible timetable by allowing for the number of time-slot to be increased. A penalty-decreaser and penalty-trader then were used to improve the solution quality. In another study, [27] demonstrated that VNS and a hybridisation with a genetic algorithm where the genetic algorithm imitated the concept of hyper-heuristic and case-based reasoning. In a recent study by [28], the great deluge algorithm was hybridised with a heuristic procedure known as the 'electromagnetic-like mechanism' within timetabling approaches where it based on particle swam optimisation. It worked by forcing the search to a promising area by dynamically changing the decay rate of the great deluge algorithm.

Table 4: The combination of weights and algorithmic approaches for the Toronto benchmark datasets (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; av. = average result)

Problem	av.	Best	{Parameter combination}	wLD	wSD	wHM
car91	5.58	5.12	{LDSDHM, Dy, REQ, EX}	0.1	0.2	0.7
car92	4.86	4.41	{LDSDHM, Dy, TW(4), AD}	0.2	0.3	0.5
ears83 I	37.27	36.91	{LDSDHM, St, TW(4), AD}	0.3	0.1	0.6
hec92 I	12.23	11.31	{LDSDHM, Dy, TW(3), AD}	0.2	0.5	0.3
kfu93	15.10	14.75	{SDHM, Dy, TW(4), EX}	0.0	0.1	0.9
lse91	12.55	11.41	{LDSDHM, Dy, REQ, EX}	0.1	0.5	0.4
pur93 I	6.42	5.87	{LDSDHM, St, TW(4), AD}	0.2	0.6	0.2
rye92	10.21	9.61	{SDHM, Dy, REQ, EX}	0.0	0.1	0.9
sta83 I	158.55	157.52	{LDSDHM, Dy, REQ, EX}	0.5	0.4	0.1
tre92	9.16	8.76	{LDSDHM, Dy, REQ, EX}	0.8	0.1	0.1
uta92 I	3.67	3.54	{LDSDHM, Dy, REQ, EX}	0.2	0.2	0.6
ute92	27.12	26.25	{LDSDHM, Dy, REQ, EX}	0.8	0.1	0.1
yor83 I	41.83	39.67	{LDSDHM, Dy, REQ, EX}	0.1	0.8	0.1

Table 5 reports the comparison of results for the thirteen problem instances of the Toronto benchmark for three different groups of approaches i.e. constructive heuristic, hyper-heuristic and other improvement. The comparisons are made with other approaches that have been published in journal articles. The comparison with constructive heuristic approaches shows that

the adaptive linear combination approach obtained one best result for sta83 I. Meanwhile, some other results of the adaptive 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 adaptive linear combination approach obtained five best results out of thirteen problem instances on hec92 I, rye92, sta83 I, ute92 and yor83 I. Note that some of the hyper-heuristic approaches may incorporate a two phase algorithm i.e. construction and improvement. The adaptive linear combination approach is purely a constructive algorithm that constructs the examination timetables using heuristic ordering. On the other hand, the comparison with other improvement approaches indicates that most of the results from the adaptive linear combination approach are far from the best results.

Table 5: Comparison of different approaches: constructive heuristics, hyper-heuristics and other improvement approaches for Toronto benchmark datasets. (ALC = our approach)

	Constructive Heuristic			Hyper-heuristic		Other Improvement			
Problem	[24]	[19]	[21]	[26]	[20]	[12]	[27]	[28]	ALC
car91	7.1	**4.97**	5.03	*4.97*	5.17	6.6	4.6	4.8	5.12
car92	6.2	4.32	**4.22**	*4.28*	4.32	6.0	3.9	4.1	4.41
ears83 I	36.4	36.16	**36.06**	35.86	*35.70*	29.3	32.8	34.92	36.91
hec92 I	**10.8**	11.61	11.71	11.85	11.93	9.2	10.0	10.73	*11.31*
kfu93	**14.0**	15.02	16.02	*14.62*	15.30	13.8	13.0	13.0	14.75
lse91	**10.5**	10.96	11.15	*11.14*	11.45	9.6	10.0	10.01	11.41
pur93 I	**3.9**	-	-	*4.73*	-	3.7	-	4.73	5.87
rye92	**7.3**	-	9.42	9.65	-	6.8	-	9.65	*9.61*
sta83 I	161.5	161.90	158.86	158.33	159.05	158.2	156.9	158.26	***157.52***
tre92	9.6	8.38	**8.37**	*8.48*	8.68	9.4	7.9	7.88	8.76
uta92 I	3.5	**3.36**	3.37	3.40	*3.30*	3.5	3.2	3.2	3.54
ute92	**25.8**	27.41	27.99	28.88	28.00	24.4	24.8	26.11	*26.25*
yor83 I	41.7	40.77	**39.53**	40.74	40.79	36.2	34.9	36.22	*39.67*

The bold, italic and underline entries indicate the best results for constructive heuristics, hyper-heuristics and other improvement approaches for the given problem instance

The weight combinations are divided into four different groups based on the heuristic contribution. The initial test of the weight combinations

reveals that there is only little difference when using different type of weight combinations with 0.1 increment. For instance, the weight combination of (0.1, 0.1, 0.8) is not very different from the weight combination of (0.2, 0.1, 0.7) in terms of the performance of solution quality since these weight combinations are almost the identical. In this case, the weight combinations are divided into four different groups i.e. high LD, high SD, high HM and balance. The group of high LD consists of weights (0.8, 0.1, 0.1), (0.7, 0.2, 0.1), (0.7, 0.1, 0.2); weights for high SD (0.1, 0.8, 0.1), (0.1, 0.7, 0.2), (0.2, 0.7, 0.1); weights for high HM (0.1, 0.1, 0.8), (0.1, 0.2, 0.7), (0.2, 0.1, 0.7) and weights for balance (0.3, 0.3, 0.4), (0.4, 0.3, 0.3), (0.3, 0.4, 0.3).

In order to see the difference in solution quality when using various top-window sizes and different groups of weight combination, a two-way analysis of variance is performed. From the statistical analysis, $F_{(31,58156)} = 18.750$ and $\rho(0.000) < 0.05$, it is clear that there are significant differences to solution quality when different top-window sizes and groups of weight combination are employed. Table 6 illustrates the effect of different top-window sizes. In most cases different top-window sizes performed significantly differently to one another. However, the top-window size 2 is not significantly different from size 3, while the size 3 is not significantly different from sizes 2 and 4. The result from the two-way analysis of variance shows that the solution quality of each group is statistically different where $\rho(0.000) < 0.05$ for each group comparison. In these circumstances, the solution quality that is tested with the weight combination from any different group of heuristics is statistically different.

Figures 1(a) and 1(b) illustrate the best performance of LDSDHM for car92 I and tre92 considering all combinations of weights. It demonstrated a pattern in the performance of best solution quality obtained for each of the combinations. By looking at the median value, when the weight value of HM is high enough then the solution quality value rapidly drop. On the other hand, whenever the weight value of HM is gradually decreased, then the solution quality also decreases progressively. Most of the peaks are obtained from the lowest weight value of HM while most of the slumps are obtained from the highest weight value of HM. This shows that the existence of the heuristic modifier in this adaptive linear combination of heuristics has an effect on 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.

Table 6: The effect of different top-window size for LDSDHM of Toronto benchmark datasets.

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 ($p = 0.001$))

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

(a)
(b)

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. It indicates that the group of highLD contributes a higher penalty value at each top-window size while the highSD, highHM and balance are almost identical in terms of penalty value during smaller size of top-window performance. However, the average performance for highSD, highHM and balance started to differ when the top-window size is 6 and above. In these circumstances, this indicates that by 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 is tested with several combinations of weights with only top-window size 3 and 5. The heuristic modifier is increased using additive and exponential with dynamic modification of the heuristic modifier value. Table 7 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 7, it reveals diverse patterns of the weight combination of each heuristic for different instances. As can be seen, about half of the problem instances obtained good quality solutions when the weight of the SD is high. Meanwhile, the weight LD and HM are varied in specific ways. Since the ITC2007 benchmark datasets are tested with only certain parameter settings, unlike the Toronto benchmark datasets, and with time limitation, these datasets might not show the exact pattern of the whole weight behaviour. Moreover, the ITC2007 benchmark datasets represent a capacitated timetabling problem and therefore, they differ from the Toronto benchmark datasets in terms of various hard and soft constraint requirements.

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 adaptive linear combination approach in this paper is presented as a purely constructive approach that iteratively constructs the examination timetable. In order to make a fair comparison, the constructed solutions are then improved using an approach introduced by [29] that employed a reheat mechanism to the great deluge algorithm. This approach is an effective method that has obtained two best results out of the twelve problem instances while other problem instances are close to the best.

As can be seen in Table 8, it shows the comparison of the best penalty values of the adaptive linear combination approach compared with other

approaches from the competition and post-competition. As can be seen, in any ways, the results of the constructed solutions of the adaptive linear combination approach cannot beat the best results obtained so far and are quite far from them. However, the constructed solutions of the adaptive linear combination approach are able to produce a 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 with [33] for Exam_10, [28] for Exam_4 and also [31] 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 [32] for Exam_10 and Exam_12, while [31] and [28] do not have solution for Exam_9, Exam_10, Exam_11 and Exam_12.

Table 7: Different combination of weights and algorithmic approaches for ITC2007 benchmark datasets (LD = largest degree; SD = saturation degree; HM = heuristic modifier; Dy = dynamic; TW = top-window; AD = additive; EX = exponential; w = weight); av. = average result

Problem	av.	Best	{Parameter combinations}	wLD	wSD	wHM
Exam_1	11 883	11 060	{LDSDHM, Dy, TW(5), EX}	0.5	0.3	0.2
Exam_2	4 021	3 133	{LDSDHM, Dy, TW(3), AD}	0.4	0.1	0.5
Exam_3	20 305	19 098	{LDSDHM, Dy, TW(3), EX}	0.1	0.7	0.2
Exam_4	21 880	21 309	{LDSDHM, Dy, TW(3), AD}	0.3	0.6	0.1
Exam_5	8 230	7 975	{LDSDHM, Dy, TW(3), EX}	0.3	0.5	0.2
Exam_6	28 985	28 330	{LDSDHM, Dy, TW(5), EX}	0.1	0.8	0.1
Exam_7	16 540	15 912	{LDSDHM, Dy, TW(5), AD}	0.1	0.8	0.1
Exam_8	20 963	20 066	{LDSDHM, Dy, TW(3), EX}	0.7	0.1	0.2
Exam_9	2 212	2 165	{LDSDHM, Dy, TW(3), AD}	0.4	0.2	0.4
Exam_10	16 912	16 516	{LDSDHM, Dy, TW(3), AD}	0.1	0.3	0.6
Exam_11	47 650	45 873	{LDSDHM, Dy, TW(3), AD}	0.1	0.6	0.3
Exam_12	8 360	7 465	{LDSDHM, Dy, TW(3), AD}	0.7	0.2	0.1

When comparing the previous results with our improved solutions, it can be seen that the method obtained promising results such as Exam_3 and Exam_11 where they are placed as the third and second best approaches. On the other hand, the results of other problem instances did not perform well.

On average, the approach is placed as the fifth best out of eleven approaches published in the literature. Note that the results are obtained with only one run as for comparison purposes, opposed to other approaches that were obtained with multiple runs.

Table 8: Comparison of adaptive linear combination approach with different approaches of ITC2007 benchmark datasets. (ALC(b) = adaptive linear combination with improvement)

Problem	[33]	[32]	[30]	[29]	[31]	[34]	[28]	ALC(b)
Exam_1	4 370	5 905	4 370	4 633	6 235	4 775	**4 368**	5 231
Exam_2	400	1 008	385	405	2 974	**385**	390	433
Exam_3	10 049	13 862	9 378	9 064	15 832	**8 996**	9 830	9 265
Exam_4	18 141	18 674	**15 368**	15 663	35 106	16 204	24 822	17 787
Exam_5	2 988	4 139	2 988	3 042	4 873	**2 929**	3 022	3 083
Exam_6	26 950	27 640	26 365	25 880	31 756	**25 740**	25 995	26 060
Exam_7	4 213	6 683	4 138	**4 037**	11 562	4 087	4 067	10 712
Exam_8	7 861	10 521	7 516	**7 461**	20 994	7 777	7 519	12 713
Exam_9	1 047	1 159	1 014	1 071	-	**964**	-	1 111
Exam_10	16 682	-	14 555	14 374	-	**13 203**	-	14 825
Exam_11	34 129	43 888	31 425	29 180	-	**28 704**	-	28 891
Exam_12	5 535	-	5 357	5 693	-	**5 197**	-	6 181

4. Conclusion

In this paper, an adaptive 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 employed 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* is scheduled first based on two strategies. This approach is tested with single and multiple heuristics with and without a heuristic modifier on Toronto while the ITC2007 benchmark datasets are tested with only multiple heuristics with heuristic modifier. The results show that by combining

multiple heuristics with a heuristic modifier, 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 constrained 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, good approximate solutions could be obtained. 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, therefore, concluded that this approach is simple and effective, and hence has potential for practical use.

References

[1] Schindl, D. (2005). Some new hereditary classes where graph coloring remains NP-hard. *Discrete Mathematics*, 295 (1-3), 197-202.

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

[3] Carter, M. W., & Laporte, G. (1996). Recent developments in practical examination timetabling. *Selected papers from the first international conference on practice and theory of automated timetabling* (pp. 3-21). Springer-Verlag.

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

[5] Burke, E. K., Elliman, D., Ford, P. H., & Weare, R. F. (1996). Examination timetabling in British universities: A survey. *Selected papers from the first international conference on practice and theory of automated timetabling* (pp. 76-90). Springer-Verlag.

[6] Burke, E. K., Kingston, J., & de Werra, D. (2004). Applications to timetabling. In J. Gross, & J.Yellen (Eds.), *Handbook of graph theory*, (pp. 445-474). Chapman Hall/CRC Press.

[7] Ersoy, E., Özcan, E., & Sima Uyar, A. (2007). Memetic algorithms and hyperhill-climbers. In P. Baptiste, G. 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*, (pp. 159-166).

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

[9] Naji Azimi, Z. (2005). Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, 163(2), 705-733.

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

[11] Abdullah, S., Ahmadi, S., Burke, E. K., & Dror, M. (2007). Investigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling. *OR Spectrum*, 29(2), 351-372.

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

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

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

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

[16] Abdul Rahim, S. K., Bargiela, A., & Qu, R. (2009). Granular modelling of exam to slot allocation. *In roceedings of the 23rd european conference on modelling and simulation (ECMS), Madrid, Spain* (pp. 861-866).

[17] Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., & Özcan, E. (2010). A classification of hyper-heuristic approaches. In M. Gendreau and J-Y Potvin (Eds.) *Handbook of metaheuristics* (pp. 449-468). Springer.

[18] Abdul Rahman, S., Bargiela, A., Burke, E. K., McCollum, B., & Özcan, E. (2009). Construction of examination timetables based on ordering heuristics. *In proceedings of the 24th international symposium on computer and information sciences* (pp. 727-732).

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

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

[21] Burke, E. K., Pham, N., Qu, R., & Yellen, J. (2010) Linear combinations of heuristics for examination timetabling. *Annals of Operations Research*, doi:10.1007/s10479-011-0854-y

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

[23] Joslin, D. E., & Clements, D. P. (1999). "Squeaky wheel" optimization. *Journal of Artificial Intelligence Research*, 10, 353-37.

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

[25] McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., & Qu, R. (2008). A new model for automated examination timetabling. *Annals of Operations Research*, 2, 2-3.

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

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

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

[29] McCollum, B., McMullan, P., Parkes, A. J., Burke, E. K., & Abdullah, S. (2009). An extended great deluge approach to the examination timetabling problem. *In proceedings of the 4th multidisciplinary international conference on scheduling: theory and applications (MISTA09), Dublin.*

[30] Müller, T. (2009). ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, 172(1), 429-446.

[31] Burke, E. K., Qu, R., & Soghier, A. (2010). 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.* (pp. 136-152).

[32] Gogos, C., Alefragis, P., & Housos, E. (2008). A Multi-Staged Algorithmic Process for the Solution of the Examination Timetabling Problem. *Practice and theory of automated timetabling (PATAT 2008), Montreal, 19-22, August 2008.*

[33] Müller, T. (2008). ITC 2007: Solver description. *Practice and theory of automated timetabling (PATAT 2008), Montreal, 19-22, August 2008.*,

[34] Gogos, C., Alefragis, P., & Housos, E. (2010). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 3, 1-3.

[35] Kahar M. N. M. & Kendall G. The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. *European Journal of Operational Research*, 207, 557-565.