MISTA 2009

Adaptive Selection of Heuristics within a GRASP for Exam Timetabling Problems

Edmund K. Burke · Rong Qu · Amr Soghier

Abstract In this paper, we describe the development of a Greedy Random Adaptive Search Procedure (GRASP) where two low-level graph heuristics, Saturation Degree (SD) and Largest Weighted Degree (LWD) are dynamically hybridised in the construction phase to construct solutions for exam timetabling problems. The problem is initially solved using an intelligent adaptive LWD and SD graph hyper-heuristic which constructs the restricted candidate list (RCL) in the first phase of GRASP. It is observed that the size of the RCL used in each iteration affects the quality of the results obtained. In addition, the SD heuristic is essential to construct a RCL which leads to a feasible solution. However, SD does not perform well at the early stages of the construction. Therefore, LWD is used until a certain switching point is reached. The hyper-heuristic adaptively determines the size of the RCL in each iteration and the best switching point after evaluating the quality of the solutions produced. In the improvement phase of GRASP, it is observed that tabu search slightly improves the constructed solutions when compared to steepest descent but it takes a longer time. The approach adapts to all the benchmark problems tested. The comparison of this approach with state-of-the-art approaches indicates that it is a simple yet efficient technique. The results also indicate that the technique could adapt itself to construct good quality solutions for any timetabling problem with similar constraints.

1 Introduction

For the past ten to fifteen years, meta-heuristics have strongly influenced the development of modern search technology at the inter-disciplinary interface of Artificial Intelligence and Operational Research [4]. Meta-heuristics can be applied in a wide variety of different areas

Edmund K.Burke
School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
E-mail: ekb@cs.nott.ac.uk

Rong Qu
School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
E-mail: rxq@cs.nott.ac.uk

Amr Soghier
School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
E-mail: azs@cs.nott.ac.uk

such as scheduling, data mining, cutting and packing, bio-informatics and many others. This facilitated the understanding of different problems in depth and the development of intelligent systems which would automatically solve large complex problems. Many of the meta-heuristic approaches which have been developed are particularly problem-specific. This is appropriate in certain circumstances but there are scenarios where we require a more generic methodology. This motivates the idea of designing automated systems that can operate on a range of related problems rather than on just one particular problem [5].

Hyper-heuristics represent a promising search technology that is concerned with improving the generality of optimisation systems.This area involves the design of intelligent systems that have the ability to decide automatically upon the appropriate methodology to be used to solve a certain problem.

University exam timetabling is an extremely important administrative activity that has attracted many researchers over forty years [22]. Many techniques over the years have been based on graph colouring [3]. This is the process of assigning colours to the vertices in a graph so that no adjacent vertices share the same colour. The objective is to use the smallest number of colours possible. Exam timetabling problems with only hard constraints can be represented as graph colouring models. Vertices in the graph represent exams in the problem, and edges represent the conflicts between exams. Colours correspond to time slots. Several heuristics (e.g. [2]) have been developed to solve graph colouring problems. Graph colouring heuristics are still widely used to develop new methods to solve timetabling problems [3].

Greedy Random Adaptive Search Procedure (GRASP) is a meta-heuristic which has attracted significant recent attention [15]. It is a two phase procedure which starts by adaptively creating a randomised greedy solution followed by an improvement phase.

In this paper, we design an intelligent adaptive hyper-heuristic within a GRASP for exam-timetabling problems. Section 2 defines some important terminology and presents background on exam timetabling, hyper-heuristics, graph colouring and GRASP. Section 3 discusses our intelligent adaptive hyper-heuristic using GRASP. The experimental settings and results are presented in section 4. Finally, conclusions are discussed and some future work is proposed in section 5.

2 Background

2.1 The Toronto Benchmark Exam Timetabling Problem

Exam timetabling [22] is the process of assigning a number of exams into a limited number of time slots taking into account a number of constraints. Constraints are divided into two types:

- *Hard Constraints*: These are constraints that cannot be violated under any circumstances.
- *Soft Constraints*: These are constraints that must be satisfied but could be violated. A penalty is added to the evaluation of a solution when these constraints are broken.

We investigate our approach by applying it to the Toronto benchmark exam timetabling problems [10]. This dataset is widely used in exam timetabling research by different state-of-the-art approaches and can therefore be considered as a benchmark [22]. The problem has one hard constraint where conflicting exams cannot be assigned to the same time slot. In addition, a soft constraint is present where conflicting exams should be spread throughout the timetable as far as possible from each other. An objective function is used to evaluate the quality of the solutions constructed. The goal here is to minimise the cost in this objective function. The quality of a solution is based on the sum of proximity costs given as:

$$\sum_{i=1}^{4} \frac{w_i \, N}{S} \qquad (1)$$

where

- $w_i = 2^{5-i}$ is the cost of assigning two exams with i time slots apart. Only exams with common students and which are five or less time slots apart are considered as violations
- N is the number of students involved in the conflict
- S is the total number of students in the problem

There has, however, been a problem with these benchmarks. Over the years two slightly different versions of some of the sets have been made available. This has caused some confusion when comparing different approaches in the literature. They were renamed version I and version II in [22] and the issues were thoroughly discussed. In the version II problem instances, 3 instances (sta83 II, yor83 II and ear83 II) contain repetitions in the data files. Therefore, these repetitions were removed [22] and named as version IIc (the “c” stands for “corrected”). A modification to the number of time slots had to be made for sta83 II from 13 to 35 time slots to obtain a feasible solution. The characteristics of the different versions are presented in table 1. The original version I, version II and the version IIc data files are all available at http://www.asap.cs.nott.ac.uk/resources/data.shtml .

Problem	Exams I/II/IIc	Students I/II/IIc	Enrolments I/II/IIc	Density	Time Slots
car91 I	682	16925	56877	0.13	35
car92 I	543	18419	55522	0.14	32
ear83 I	190	1125	8109	0.27	24
ear83 IIc	189	1108	8057	0.27	24
hec92 I	81	2823	10632	0.42	18
hec92 II	80	2823	10625	0.42	18
kfu93 I	461	5349	25113	0.06	20
lse91 I	381	2726	10918	0.06	18
sta83 I	139	611	5751	0.14	13
sta83 IIc	138	549	5417	0.19	35
tre92 I	261	4360	14901	0.18	23
uta92 I	622	21266	58979	0.13	35
uta92 II	638	21329	59144	0.12	35
ute92 I	184	2750	11793	0.08	10
yor83 I	181	941	6034	0.29	21
yor83 IIc	180	919	6002	0.3	21

Table 1 Characteristics of the Toronto Benchmark dataset

2.2 Greedy Random Adaptive Search Procedure (GRASP)

Randomly generated solutions are not expected to be of high quality. A greedy function, which does not contain ties or which contains deterministic rules to break ties, will always lead to the same solution. To overcome such issues a semi-greedy heuristic such as GRASP can be used [14, 17]. After the candidate list is produced randomly, the best ranked elements are placed in a restricted candidate list. The element to add to the solution is then randomly chosen. Two different methods were proposed by [14] and [17] to choose the elements to place in the restricted candidate list (RCL). The cardinality-based method chooses a number of elements from the top of the list. The size of the RCL is set through a parameter. The value-based method chooses the elements that fall within a certain threshold (0-100%) from the greedy value. More methods were also proposed in [17].

GRASP can be easily integrated in a hybrid meta-heuristic, as any local search algorithm could be used in the improvement phase. Simulated Annealing and Tabu Search have been used in [11, 12, 19] as local search procedures.

2.3 Hyper-Heuristics

The idea of Hyper-heuristics is to decide the appropriate search method to be used to solve a certain problem. A hyper-heuristic explores the heuristic space instead of exploring the solution space. After choosing certain heuristics to be used, this heuristic combination can be run on the problem and the solution can be evaluated using an objective function. The hyper-heuristic can then use this evaluation to decide whether to accept the current technique or switch to a different one.

Several hyper-heuristics have been recently developed to solve many combinatorial optimisation problems, which include timetabling [6, 7]. Low-level heuristics as employed within hyper-heuristic methods in the literature can be divided into two main categories: Constructive and Improvement.

2.3.1 Hyper-heuristics with Constructive Low-level Heuristics

A hyper-heuristic with low-level constructive heuristics starts with no solution and builds it step by step from the beginning. For example, a hyper-heuristic to solve an exam timetabling problem could initially start with an empty timetable. It then starts creating a full timetable by choosing a certain exam and inserting it in the timetable using a certain heuristic. The task of the hyper-heuristic might be to create a heuristic list which is the combination of heuristics that would allocate all the exams in appropriate time slots. Different heuristic sequences could be created and applied to the problem at hand until all the exams are scheduled and the solution is accepted. During the timetable construction, the hyper-heuristic evaluates the moves and chooses the next heuristic to be applied. In certain cases, the hyper-heuristic could decide to remove exams already scheduled or back track according to the evaluation carried out in each step.

In the recent timetabling literature, relatively few constructive techniques have been investigated compared to improvement techniques (see [7]). A technique employing fuzzy logic to combine graph heuristics to solve exam-timetabling problems was discussed in [1]. In [24], an investigation is made to employ Genetic Algorithms to evolve the configurations of constraint satisfaction methods on constructing problem solutions.

Another approach was developed in [9] where a Case-Based Reasoning system was implemented. The system is used as a heuristic selector for solving exam timetabling problems. Different problem solving situations and their corresponding constructive heuristics are stored in the system. The system then compares the problem situation at hand with the cases that are already stored. The heuristics previously used are retrieved to solve the problem.

In [23], a genetic-based hyper-heuristic is employed using four basic constructive low-level heuristics to solve one-dimensional bin packing problems. Very good results have been found and some potential issues were discussed.

2.3.2 Hyper-heuristics with Low-level Improvement Heuristics

This Hyper-heuristic starts with a complete timetable, which it will then iteratively improve. Several heuristics could then be applied such as moving an exam from one slot to another or swapping two exams. The effect of applying a certain heuristic is then examined and the hyper-heuristic decides whether to approve the move or to perform a different one.

In [18], a tabu list was used to avoid the repetition of bad moves for a certain number of iterations. In [6], a similar approach was applied to several instances of a nurse rostering and a course-timetabling problem. It was shown to be competitive with algorithms specifically designed for these problems.

The tabu search hyper-heuristic investigated in [6] inspired the development of a Simulated Annealing method to search a set of low-level heuristics to determine the shipper sizes in transportation problems in [13]. The low-level heuristics here are both neighbourhood structures and sampling policies within the solution space.

In [16], a Genetic Algorithm is used to choose the low-level heuristics to be applied to simulated course scheduling and student project presentation problems. The low-level heuristics employed here represent some move strategies that are applied to the problems.

2.4 The Graph based Hyper-heuristic (GHH)

Several heuristics have been developed to solve graph colouring problems and exam timetabling problems (e.g. [10]). The idea of these heuristics is based on estimating difficulty and colouring the most difficult vertices first in a simple constructive technique. Some graph heuristics that have been used in timetabling research can be presented as follows (These descriptions are taken from [10]):

– *Largest Degree First (LD): Exams that have a larger number of clashes (common students) with other exams should be scheduled first.*
– *Largest Colour Degree First (CD): Exams that have a larger number of clashes with the exams already scheduled should be scheduled first.*
– *Saturation Degree (SD): This heuristic will order the exams according to the number of time slots available for this exam to be scheduled. The exams with the lowest number of available time slots will be given higher priority.*
– *Largest Weighted Degree (LWD): This heuristic is similar to LD but uses more information when applied to exam-timetabling problems. It orders the exams that have the largest number of students, involved in clashes with other exams first.*
– *Largest Enrolment (LE): Largest enrolment orders the exams according to the number of students in each exam.*

In a graph based hyper-heuristic for timetabling problems [20], a sequence of graph heuristics are used to order the events (courses or exams) one by one to construct a full timetable. A number of high-level heuristics were used to compare their performance to find the best heuristic sequences to solve exam and course timetabling problems. In each iteration, a heuristic sequence is chosen and used to construct a solution. After a sequence is applied and a solution is generated, this solution is evaluated using the objective function.

A timetable is constructed by adding the events one by one in the available time slots. According to the heuristic sequence being followed, the first event in the ordered list is scheduled to the first time slot that leads to the least cost. The cost in this situation is calculated by the soft constraint violations incurred. After an event is scheduled, the remaining events are re-ordered by using the next heuristic in the heuristic sequence. The process continues until all the events are scheduled and a full timetable is constructed.

An illustrative example of ordering five exams (e1-e5) using two graph heuristics is shown in figure 1. The example assumes that the events are ordered according to their SD and LWD as shown. In the 1^{st} and 2^{nd} iterations, LWD is used (as shown in the heuristic sequence) and therefore e1 and e2 are scheduled. Assuming that the order of the remaining elements does not change after the SD list is updated in the 3^{rd} iteration, e5 is then scheduled. Finally, e4 followed by e3 are scheduled using the LWD ordering.

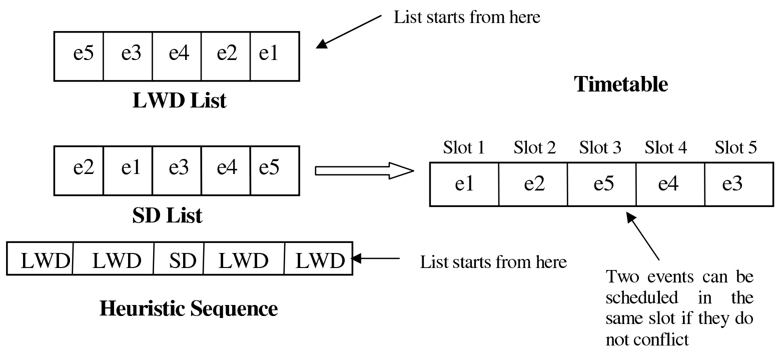

Fig. 1 An illustrative example of using a heuristic sequence to construct a timetable

The quality of the timetable constructed is then evaluated and the cost is compared with any previous timetables produced using other sequences. Certain sequences during the construction of the timetable cannot schedule a certain event as it causes conflicts. Hence, the sequence is discarded and the high-level heuristic uses another sequence to construct another solution. In this approach, the high-level heuristic acts on the heuristic sequences and does not act on the problem domain itself. The search is performed on the heuristic space and the high-level heuristic is controlled by the performance of running heuristic sequences to construct timetables. The GHH approach is used to construct a heuristic sequence which will then construct a solution. This technique applies heuristics adaptively to construct solutions and could therefore be generalised to be applied to a set of similar problems (i.e. timetabling problems in this case). In this paper, we present an adaptive approach where two low-level graph heuristics (SD and LWD) are dynamically hybridised in the construction phase of a GRASP. The hyper-heuristic uses SD & LWD to construct the RCL in the GRASP. The exam to be scheduled next is randomly chosen from the RCL. The process continues until all the exams are scheduled or a certain exam causes a conflict. Furthermore, an improvement is made to the constructed timetables in the second phase of GRASP.

3 An Intelligent Adaptive GHH within a GRASP

3.1 An Intelligent Adaptive Hybridisation of SD and LWD within a GRASP

As previously stated, a random ordering method was used in an iterative approach to generate heuristic sequences to be applied on exam and course timetabling problems [20]. Applying random heuristic sequences to a problem is usually a time consuming process. Therefore, an intelligent approach that would always lead to a feasible solution would be preferred. Figure 2 presents the pseudo-code of an intelligent technique to hybridise the SD and LWD graph heuristics using a GRASP.

```
for n = 0 to n = 50                // n : number of iterations
{
        while(number of exams scheduled  <  total number of exams)
        {
                RclSize = 0.03 x u            // u : number of unscheduled exams
                Create Heuristic Sequence & restricted candidate list of RclSize
                Fill two thirds of the heuristic sequence with the SD heuristic
                Starting from the switching point, fill the rest of the heuristic sequence with LWD
                Run the heuristic sequence to the problem and fill the RCL
                Schedule any exam randomly from the RCL
        }

        if(solution  is feasible)
                Evaluate the solution and apply a steepest descent improvement
        else
        {
                Discard the solution & continue without applying an improvement
        }
        //end if

        if(current solution is better than the best solution)
                best solution = current solution
        //end if
}
```

Fig. 2 The pseudo-code of the intelligent hybridised LWD and SD graph based hyper-heuristic within GRASP

It was observed in our previous work [21] that heuristic sequences that combine SD with LWD perform the best for almost all the exam timetabling problem instances that were investigated. The motivation of intelligently hybridising LWD and SD comes from the fact that LWD identifies the difficulty of scheduling exams in relation to the rest and SD orders them according to the current state of the available time slots. Therefore, we developed a hyper-heuristic that combines SD and LWD within the construction phase of a GRASP. The GHH is applied to the problem at hand to create a restricted candidate list (RCL) which is used to randomly choose the next exam to be scheduled.

In each iteration, the GHH starts by updating the SD and LWD orderings. The GHH then determines the size of the heuristic sequence required to construct the RCL. The size was initially set to 3% of the number of unscheduled exams. Different RCL sizes were tested and it was found that using a RCL which has a size of 1%-3% of the unscheduled exams obtains the best solutions according to the size of the problem. Using a RCL of a large size increases the randomness in the solution construction and therefore the choice of the exam to be scheduled next becomes more difficult. In contrast, using a very small RCL would limit the choice of exams and would not lead to a feasible solution. After determining the size of the heuristic sequence, the heuristics to be used are defined. The sequence is run on the problem at hand and the outcome exams are placed in the candidate list.

According to the calculated size of the RCL, the GHH creates the heuristic sequence to construct the RCL. In previous work undertaken in [7, 8, 10], it was proved that using SD on its own outperforms other graph heuristics and results in a better outcome. This is due to the ability of SD to dynamically adapt itself to the current state of the solution construction as it orders events according to the available valid time slots. Therefore, we decided to let the GHH to construct a heuristic sequence which contains twice as many SD iterations than LWD. Hence, this technique gives the GRASP a higher chance to schedule the exams using SD. The size of the RCL will decrease adaptively as the exams are scheduled. The approach will also automatically schedule the last 20 exams using SD only. In addition, an exam that is on top of the SD and LWD orderings will be placed in the RCL twice and will have a higher chance of being randomly chosen and scheduled.

An illustrative example is given in figure 3 to use a GHH to construct a RCL. The GHH uses SD and LWD as low-level heuristics. The number of unscheduled exams are equal to 100. Hence, a heuristic sequence and RCL of size 3 are created. The sequence generated will contain two SD iterations and one LWD as the number of SD elements have to be twice as many as the LWD elements in the RCL. The heuristic sequence is run and e56 & e6 will be taken from the SD list and placed in the RCL. In addition, e9 will be taken from the LWD list and placed in the RCL. Therefore, the RCL will contain a ratio of 2:1 SD against LWD elements.

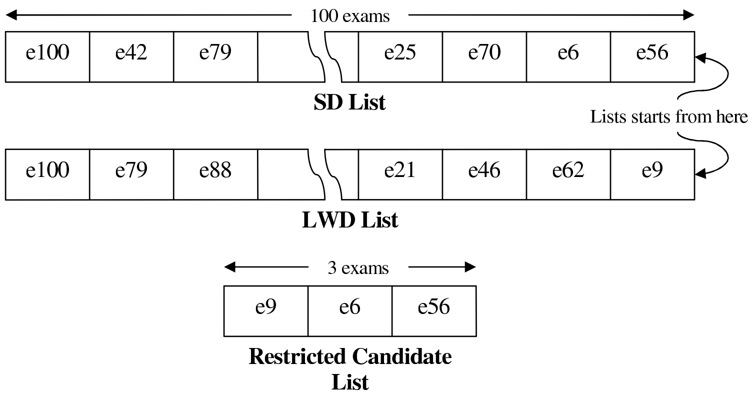

Fig. 3 An illustrative example of building a RCL from SD and LWD

3.2 GRASP & GHH

GRASP is introduced here to find a quick solution that would lead to feasible solutions in the heuristic space. This is done through two steps. The adaptively hybridised LWD & SD

GHH discussed in section 3.1 is used to build the restricted candidate list in the construction phase of GRASP. Subsequently, a random exam is chosen from the RCL and scheduled. This process continues until all the exams are scheduled. Finally, the constructed solution is improved through a local search in the solution space.

3.2.1 GRASP construction phase

In this construction phase, the GHH starts by applying LWD to schedule the first exams in the first iterations. This process is done adaptively to find the switching point which leads to the best solution. The GHH then switches to combine LWD and SD to build the GRASP RCL for the rest of the iterations. A random exam is then chosen from the RCL and scheduled. This process will either schedule all the exams and construct a feasible solution or it will stop at a certain point where the exam cannot be scheduled in any time slot due to its conflict with those already scheduled. If a feasible solution is obtained, the solution is evaluated using the objective function. An improvement is then applied to the constructed solution in the second phase of the GRASP. Otherwise, the process restarts if a feasible solution cannot be obtained.

Since SD orders exams according to the number of time slots available, starting to solve a problem using SD is inefficient as all the time slots would be available [7,8,10]. This would result in many ties at the beginning and different approaches could be used to overcome this difficulty.

In view of the fact that LWD orders the exams that have the largest number of students involved in clashes with other exams first, LWD gives priority to the exams that will be difficult to schedule later on in the process. Therefore, we used LWD to order exams in the first iterations before introducing SD.

The hybridised GHH starts by employing LWD. At a certain point, the hyper-heuristic has to switch from LWD to using SD and LWD as described in section 3.1.

We applied the intelligent GHH to the Toronto benchmark exam timetabling problems [10] to adaptively select the best switching point from using LWD to combining both LWD & SD in a GRASP. The following formula was used to choose an initial switching point which is then adaptively tuned:

$$\text{Switching Point} = \frac{\text{Total Number of Exams}}{\text{Total Number of Time slots}} \tag{2}$$

The starting switching point is the number of exams required so that the exams are spread evenly in the time slots available. The hyper-heuristic uses this as the initial switching point and applies the approach described in the previous section to obtain a solution. The switching point is then incremented and decremented by one and the new switching points are used to obtain another set of solutions. The process stops when a feasible solution is not obtained for two consecutive iterations.

Using this adaptive approach, the best switching point is automatically found for different problems. We observed that using LWD on its own will not construct a feasible solution. Furthermore, SD has to be used early in the process to be able to identify exams that will be difficult to schedule later on. In addition, it is observed that using different switching points leads to different solutions. This is due to the fact that SD dynamically orders the exams and therefore the amount of LWD iterations employed at the beginning will affect the decisions made by SD.

3.2.2 GRASP improvement phase

The GRASP improvement phase starts with a feasible solution from the construction phase and a steepest descent hill climbing improvement is employed. The hill climbing is run for $(e \times 10)$ iterations where e is the number of exams. In each iteration a random exam is chosen and the improvement of scheduling this exam in all the time slots of the timetable is observed. The best improvement is stored at the end of the process and the result is obtained.

4 Experimental Results

The adaptive intelligent GHH using GRASP was implemented and tested on a PC with an Intel Pentium 3.4 GHz processor, 1GB RAM and Windows XP. The program was coded using C++. The intelligent hyper-heuristic, the low-level graph heuristics and the GRASP are implemented as objects with a common interface to work together. The hyper-heuristic could be easily applied to different exam timetabling problems in the dataset. The objectives of the experiments are:

– To demonstrate the role of adding a GRASP to a GHH to improve the quality of the solutions obtained (shown in table 4).
– To raise the generality of exam timetabling problem solving approaches through producing a dynamic hyper-heuristic that would adapt to a given exam timetabling problem and generate results which are comparable to the results published in the literature (shown in table 4).
– To compare the quality of this approach with other known published methods (shown in table 4).

4.1 Analysis on the Intelligent Adaptive Hybridisation of the LWD and SD GHH to construct a RCL in a GRASP

In the first set of experiments, the GHH using GRASP is applied to version I and version II of the Toronto benchmark exam timetabling problems to see its ability to construct a feasible solution as well as to evaluate the quality of the solutions produced. The effect of intelligently hybridising different low-level heuristics in a GRASP is investigated. It is observed that SD and LWD performed the best when hybridised together in the RCL rather than using them separately. The GHH chooses the exams which have a high priority in both the SD and LWD orderings to construct the RCL (i.e. the exams that have the largest number of students, involved in clashes with other exams and the exams that have the lowest number of slots available at that time to schedule in).

In the second set of experiments, using different percentages of SD and LWD in the RCL was investigated. It was observed that using the SD heuristic more than the LWD in the RCL produces better results. Therefore, we used a ratio of 2:1 SD to LWD heuristics to construct the RCL. This shows the importance of SD being used as it updates the ordering in each iteration and adapts to the current situation.

Five runs with 100 iterations each using distinct seeds are carried out for all the datasets except for (car92 I), (uta92 I), (uta92 II) and (car91 I) where only three runs were performed.The average costs of the solutions and the average percentage of feasible solutions constructed in the runs are presented in table 2. A comparison with using the same approach with SD and LWD separately is also shown.

Problems	GRASP Construction using SD & LWD			GRASP Construction using only SD			GRASP Construction using only LWD		
	Average Cost	Best Cost	Percentage of feasible solutions	Average Cost	Best Cost	Percentage of feasible solutions	Average Cost	Best Cost	Percentage of feasible solutions
hec92 I	13.00	12.41	35%	13.82	13.02	21%	-	-	0%
hec92 II	13.72	12.37	20%	-	-	0%	-	-	0%
sta83 I	167.31	163.63	19%	174.61	169.10	15%	172.52	171.49	38%
sta83 IIc	40.42	36.82	69%	40.53	38.03	63%	36.03	34.79	69%
yor83 I	43.38	42.54	5%	45.62	44.21	4%	-	-	0%
yor83 IIc	52.57	52.24	3%	-	-	0%	-	-	0%
ute92 I	31.90	30.01	13%	32.78	30.74	10%	-	-	0%
ear83 I	39.06	37.71	32%	45.74	43.94	29%	-	-	0%
ear83 IIc	42.54	40.49	6%	48.27	48.21	3%	-	-	0%
tre92 I	9.38	9.00	39%	10.16	9.30	22%	-	-	0%
lse91 I	12.89	12.29	3%	13.48	12.51	2%	-	-	0%
kfu93 I	17.24	16.84	6%	18.28	18.24	2%	-	-	0%
car92 I	4.84	4.74	3%	5.07	4.85	0%	-	-	0%
uta92 I	3.65	3.62	100%	3.97	3.70	32%	-	-	0%
uta92 II	3.72	3.60	20%	-	-	0%	-	-	0%
car91 I	5.59	5.48	94%	5.78	5.49	57%	-	-	0%

Table 2 Average results, best results and percentage of feasible solutions produced from the GRASP construction phase using SD & LWD in comparison to using them separately. A (-) indicates that no feasible solution could be obtained

Furthermore, another set of experiments is run to observe the effect of changing the size of the restricted candidate list used. Different RCL sizes were tested on problems with different size. The experiments showed that using a RCL with 3% the size of the unscheduled exams produces the best results for small problems (i.e problems with less than 200 exams). For medium sized problems, a RCL with 2% the size of unscheduled exams produced the best results (i.e problems with exams between 200 - 400). For large problems, a 1% RCL produced the best results (i.e. problems with more than 400 exams). A large RCL introduces more randomness in large problems while a small RCL restricts the choice in small problems. In all the cases, the process starts with high randomness which adaptively decreases and becomes more deterministic as exams are scheduled.

Finally, another important factor for this method to construct a feasible solution is the switching point from using LWD to combining both SD and LWD in the GRASP. It is observed that for each problem, more than one switching point between the 1st and the 25th iteration led to a feasible solution. The choice of the switching point that would lead to the best solution is difficult especially when the method is applied to different problems. Using the formula and approach described in section 3.2.1, the switching point problem was adaptively solved. The average percentage of feasible solutions obtained in the runs for each problem is shown in table 2.

4.2 Analysis on GRASP improvement using steepest descent

In the second set of experiments, steepest descent is applied to the constructed solutions to perform a local search on the solution space. The improvement is applied to all the instances where a feasible solution is constructed. It is observed that the solutions with the least cost from the construction phase do not always lead to the best improvements. This is because the solutions constructed by different heuristic sequences in the heuristic space lead to different neighbourhoods in the solution space being searched to find the global optimum. We also investigated using tabu search and found that the tabu search improvement performs slightly better than steepest descent in some problems. However, the computational time for performing a tabu search improvement is much higher. Therefore, we used steepest descent as it produces competitive results in comparison with tabu search in a shorter time. Table 3 shows the average costs after the improvement and the total average time including construction and improvement.

Problems	GRASP construction Average Cost	GRASP Improvement using Steepest Descent		Total Average Time (sec)
		Average Cost	Best Cost	
hec92 I	13.00	12.14	**11.78**	83.13
hec92 II	13.72	13.51	**12.04**	267.50
sta83 I	167.31	166.39	**158.94**	314.60
sta83 IIc	40.42	38.26	**35.19**	3376.87
yor83 I	43.38	42.85	**42.19**	484.58
yor83 IIc	52.24	51.53	**50.40**	410.77
ute92 I	31.90	31.14	**26.62**	402.50
ear83 I	39.06	38.71	**36.52**	5289.05
ear83 IIc	42.54	42.10	**39.93**	737.30
tre92 I	9.38	9.31	**8.99**	10817.01
lse91 I	12.89	12.72	**12.12**	768.33
kfu93 I	17.24	16.64	**15.45**	748.32
car92 I	4.84	4.67	**4.45**	8710.55
uta92 I	3.65	3.62	**3.50**	99053.51
uta92 II	3.72	3.69	**3.49**	95687.27
car91 I	5.59	5.48	**5.37**	99874.59

Table 3 Average costs before and after improvement, best costs after improvement and total average time using Steepest Descent in GRASP

4.3 Comparison with state-of-the-art approaches

The best results obtained, the best reported in the literature using different constructive methods [22], the graph based hyper-heuristic [7], the tabu search hyper-heuristic [18] and other methods are presented in table 4. In addition, the standard deviation from the best reported results obtained is shown (σ in table 4).

The results obtained by this approach when compared to the state-of-the-art approaches, indicate the efficiency and generality of the intelligent adaptive hybridisation of LWD and SD GHH. In comparison with the pure GHH approach in [7], the GRASP-GHH performs better in 9 out of the 11 version I cases. Furthermore, it performs better in 7 out of 8 cases reported in [18]. The results of (sta83 II), (yor83 II) and (ear83 II) were not compared to other results in the literature since the methods of modification to deal with the corruption (i.e. repetitions) in the datasets were not stated.

Although the best results reported in the literature were obtained by using different approaches that worked well only on specific instances, the results produced here using the adaptive approach are still in the range of the best-reported results as shown. In addition, the aim of this paper is to illustrate the adaptiveness and the effect of hybridising low-level heuristics in a hyper-heuristic using GRASP that could be applied to any timetabling problem with similar constraints. The objective here is not to outperform the other approaches that worked only on specific instances. However, we can demonstrate that our approach is competitive to the other approaches in the literature. It was not possible to compare the computational time as the time for most of the approaches was not reported in the literature.

Problems	Steepest Decent in GRASP		GHH Best [7]	Tabu search HH [18]	Constructive Best [22]	Best Reported [22]
	Best Cost	σ				
hec92 I	11.78	1.82	12.72	11.86	10.8	**9.2**
hec92 II	12.04	0.88	-	-	-	10.8
sta83 I	158.94	1.16	158.19	157.38	158.19	**157.3**
sta83 IIc	35.19	-	-	-	-	-
yor83 I	42.19	4.24	40.13	-	39.80	**36.20**
yor83 IIc	50.40	-	-	-	-	-
ute92 I	26.62	0.54	31.65	27.60	25.8	**24.4**
ear83 I	36.52	5.11	38.19	40.18	36.16	**29.3**
ear83 IIc	39.93	-	-	-	-	-
tre92 I	8.99	0.78	8.85	8.39	8.38	**7.9**
lse91 I	12.12	1.78	13.15	-	10.50	**9.6**
kfu93 I	15.45	1.73	15.76	15.84	14.00	**13.0**
car92 I	4.45	0.37	4.84	4.67	4.32	**3.93**
uta92 I	3.50	0.25	3.88	-	3.36	**3.14**
uta92 II	3.49	0.13	-	-	-	3.40
car91 I	5.37	0.62	5.41	5.37	4.97	**4.5**

Table 4 Best results obtained by the intelligent adaptive GHH using GRASP compared to the state-of-the-art approaches

5 Conclusions and Future Work

In this paper, we present an adaptive approach where two low-level graph heuristics (saturation degree and largest weighted degree) are dynamically hybridised in the construction phase of a greedy random adaptive search procedure (GRASP) for exam timetabling problems. The problem is initially solved using an intelligent adaptive LWD & SD graph hyper-heuristic which constructs the restricted candidate list in the construction phase of GRASP. It is observed that the SD heuristic is essential to construct a feasible solution. However, SD does not perform well at the early stages of the construction as the timetable is empty and it returns the same difficulty value for scheduling exams at the first few iterations. Therefore, LWD is used until a certain switching point. The switching point is adaptively tuned by the hyper-heuristic after evaluating the quality of the solutions generated.

An improvement is made to the constructed solutions and the best is chosen. Steepest descent is used for the improvement and it is shown to produce competitive results to a tabu search improvement in a shorter time. It is observed from experiments that the approach adapts to all the problems it is applied to and generates different quality solutions according to the switching point chosen. Therefore, we created an adaptive adjustment to the switching point according to the quality of the solutions produced.

The comparison of this approach with state-of-the-art approaches indicates that it is a simple yet efficient technique. It is also a more general technique than many in the literature.

It can be adapted to construct good quality solutions for any exam timetabling problem with similar constraints.

Future work will include intelligently hybridising more different low-level graph colouring heuristics. In addition, some new adaptive heuristics similar to SD could be designed. Considering the cost of assigning an exam in a certain time slot instead of the first available slot could also improve the quality of the constructed solutions. Tie breakers for the ordered exams could also be considered as well as different switching point selectors.

Finally, one of our primary future research directions is applying the technique developed here to random exam timetabling datasets and evaluating its ability to adapt to the problems being solved. Furthermore, the technique developed could be applied to graph colouring and course-timetabling problems.

References

1. H. Asmuni, E.K. Burke, J. Garibaldi, and B. McCollum. Fuzzy multiple ordering criteria for examination timetabling. In E.K. Burke and M. Trick, editors, *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, volume 3616 of *Lecture Notes in Computer Science*, pages 334–353. Springer, 2004.
2. D. Brelaz. New methods to colour the vertices of a graph. *Communications of the ACM*, 22:251–256, 1979.
3. E.K. Burke, D. de Werra, and J. Kingston. Applications to timetabling. In J. Gross and J. Yellen, editors, *Handbook of Graph Theory*, chapter 5.6, pages 445–474. Chapman Hall/CRC Press, 2004.
4. E.K. Burke and G. Kendall, editors. *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*. Springer, 2005.
5. E.K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. Kochenberger, editors, *Handbook of Meta-Heuristics*, pages 457–474. Kluwer, 2003.
6. E.K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, Dec 2003.
7. E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176:177–192, 2007.
8. E.K. Burke and J. Newall. Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of operations Research*, 129:107–134, 2004.
9. E.K. Burke, S. Petrovic, and R. Qu. Case based heuristic selection for timetabling problems. *Journal of Scheduling*, 9(2):115–132, 2006.
10. M.W. Carter, G. Laporte, and S.Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of Operational Research Society*, 74:373–383, 1996.
11. R. Colome and D. Serra. Consumer choice in competitive location models: Formulations and heuristics. Technical report, Department of Economics and Management, Universitat Pompeu Fabra, Barcelona, Spain, 1998.
12. H. Delmaire, J.A. Diaz, E. Fernandez, and Ortega M. Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *INFOR*, 37:94–225, 1999.
13. K. Dowsland, E. Soubeiga, and E.K. Burke. A simulated annealing hyperheuristic for determining shipper sizes. *European Journal of Operational Research*, Vol 179, issue 3:pp 759–774, 2007.
14. T. Feo and M. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.
15. T. Feo and M. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
16. L. Han and G. Kendall. Guided operators for a hyper-heuristic genetic algorithm. In T. D. Gedeon and L. C. C. Fung, editors, *Proceedings of AI-2003: Advances in Artificial Intelligence. The 16th Australian Conference on Artificial Intelligence (AI 03)*, pages 807–820, 2003.
17. J.P. Hart and A.W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6:107–114, 1987.
18. G. Kendall and N. Mohd Hussin. An investigation of a tabu search based hyper-heuristic for examination timetabling. In G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, editors, *Selected Papers from MISTA 2003*, pages 309–328. Springer, 2005.

19. M. Laguna and Gonzalez-Velarde. A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent manufacturing*, 2:253–260, 1991.
20. R. Qu and E.K. Burke. Hybridisations within a graph-based hyper-heuristic framework for university timetabling problems. *Journal of Operational Research Society*, 60:1273–1285, 2009.
21. R. Qu, E.K. Burke, and B. McCollum. Adaptive automated construction of hybird heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, 198(2):392–404, 2009.
22. R. Qu, E.K. Burke, B. McCollum, L.T.G. Merlot, and S.Y. Lee. A survey of search methodologies and automated approaches for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009.
23. P. Ross, J.G. Marin-Blazquez, S. Schulenburg, and E. Hart. Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03),*, pages 1295–1306, 2003.
24. H. Terashima-Marin, P. Ross, and M. Valenzuela-Rendon. Evolution of constraint satisfaction strategies in examination timetabling. In *Proceedings of the Genetic and Evolutionary Conference, Orlando, Florida*, pages 635–642, 1999.