Appl Intell (2012) 37:1–11
DOI 10.1007/s10489-011-0309-9

A graph coloring constructive hyper-heuristic for examination timetabling problems

Nasser R. Sabar · Masri Ayob · Rong Qu · Graham Kendall

Published online: 3 August 2011
© Springer Science+Business Media, LLC 2011

Abstract In this work we investigate a new graph coloring constructive hyper-heuristic for solving examination timetabling problems. We utilize the hierarchical hybridizations of four low level graph coloring heuristics, these being largest degree, saturation degree, largest colored degree and largest enrollment. These are hybridized to produce four ordered lists. For each list, the difficulty index of scheduling the first exam is calculated by considering its order in all lists to obtain a combined evaluation of its difficulty. The most difficult exam to be scheduled is scheduled first (i.e. the one with the minimum difficulty index). To improve the effectiveness of timeslot selection, a roulette wheel selection mechanism is included in the algorithm to probabilistically select an appropriate timeslot for the chosen exam. We test our proposed approach on the most widely used uncapacitated Carter benchmarks and also on the recently introduced examination timetable dataset from the 2007 International Timetabling Competition. Compared against other methodologies, our results demonstrate that the graph coloring constructive hyper-heuristic produces good results and outperforms other approaches on some of the benchmark instances.

Keywords Examination timetabling · Graph coloring · Hybridization · Hyper-heuristics · Roulette wheel selection

N.R. Sabar (✉) · M. Ayob
Data Mining and Optimisation Research Group (DMO), Centre for Artificial Intelligent (CAIT), Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia
e-mail: naserdolayme@yahoo.com

M. Ayob
e-mail: masri@ftsm.ukm.my

R. Qu · G. Kendall
ASAP Research Group, School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK

R. Qu
e-mail: rxq@cs.nott.ac.uk

G. Kendall
e-mail: Graham.Kendall@nottingham.ac.uk

G. Kendall
University of Nottingham Malaysia Campus, 43500 Semenyih, Selangor, Malaysia

1 Introduction

Educational timetabling is an ongoing challenge that most academic institutions face when scheduling courses or exams. This is due to the large number of constraints that have to be accommodated. Courses are often scheduled by the individual faculties and departments, whereas the examination timetable is usually centrally generated to cover the entire university. Both problems are complex, involving a variety of constraints, and thus present a challenging topic for both researchers and practitioners.

Examination timetabling can be defined as the process of assigning a set of exams into a limited number of timeslots and rooms so as not to violate any hard constraints and to minimize soft constraint violations as much as possible [36]. *Hard constraints* have to be respected in order to have a feasible timetable. For example, no student can sit more than one exam at the same time and there must be a sufficient number of seats to accommodate the exams being scheduled in a given room. A *soft constraint* represents a constraint that, ideally, should be satisfied as far as possible. However, the timetable is still considered feasible even if some of these soft constraints are violated. An example of a soft constraint is that exams should be spread within the timetable for a given student, and the aim is to minimize

soft constraint violations. Therefore, the timetabling problem can be considered as a problem of minimizing soft constraint violations, while respecting all the hard constraints.

A large number of different approaches have been developed for solving examination timetabling problems in the last four decades. These include graph based sequential techniques [7, 39], constraint based techniques [28, 29], local search methods including tabu search [14], simulated annealing [12, 17, 18, 40], population based algorithms including genetic algorithms [37], ant colony optimization [15], scatter search [22], pattern recognition based method [21] and hybrid approaches [1, 38], etc. For more details please refer to [36]. Most of these methods aimed to develop problem specific techniques that are able to produce the best results for one or more datasets [6, 9].

Recently, there has been a growing trend toward more general methods. Hyper-heuristics represent one of these approaches [6, 9, 11]. The term hyper-heuristic refers to an approach that focuses on a search space of heuristics rather than a search space of solutions [7, 9, 34]. Low level heuristics (e.g. different neighborhood move structures or different constructive heuristics) are controlled by high level general mechanisms (e.g. meta-heuristics or reinforcement learning) in order to provide solutions to a wider variety of problems, rather than developing tailor-made solutions for each problem encountered [9, 24].

However, Qu and Burke [34] commented that a small change to the heuristic list (especially at the beginning) often results in quite different solutions being generated. Thus, as long as the high level search is diversified, a simple multi-start local search works as well as a fine tuned tabu search when single heuristics are used in heuristic lists in their graph based hyper-heuristics (GHH).

In GHH, as only single heuristics are used in heuristic lists, a lot of ties may occur when ordering (and selecting) exams and assigning them to timeslots during solution construction. By simply randomly choosing an exam from those of the same rank, potentially good solutions may be missed. Also by assigning a chosen exam to the first least-cost timeslot, only a small part of the solution search space can be explored by using the simple heuristic lists.

Therefore, in this work, we introduce a new graph coloring constructive hyper-heuristic (GCCHH) which utilizes the hybridizations of four graph coloring heuristics in constructing four ordered lists of exams in timetable construction. GCCHH employs hierarchical heuristics and probabilistic timeslot selection. The focus is not to design another high level search but to investigate more intelligent criteria of exam ranking and timeslot selection. The latter is seldom studied in timetabling research. More potential solutions, of higher quality, in the solution space can thus be found. Instead of sequentially applying low level heuristics to construct timetables, as has been used in previous research [7, 34], the hybridizations of the four low level heuristics are applied simultaneously. Four heuristic hybridizations, consisting of different graph coloring heuristics, have been developed and tested on the un-capacitated (where the size of the room is disregarded) Carter benchmarks (Toronto *b*, see [36]) and the ITC 2007 [23, 26] examination timetabling datasets.

The paper is organized as follows: Sects. 2 and 3 review related work on hyper-heuristics and the ITC 2007 examination timetabling, respectively. Our proposed GCCHH approach is presented in Sect. 4, followed by our results in Sect. 5. Finally discussions and concluding remarks are presented in Sects. 6 and 7.

2 Related work on hyper-heuristics for examination timetabling problems

Recently, hyper-heuristics have been attracting increasing research attention. Burke et al. [9] define a hyper-heuristic as: "*A search method or learning mechanism for selecting or generating heuristics to solve computational search problems*". The high level mechanism of the hyper-heuristic, at each iteration, selects the appropriate low level heuristic based on certain selection criteria. The high level mechanism can be, for example, any kind of meta-heuristic algorithm; whilst the low level heuristics can be, for example, constructive heuristics, graph heuristics or a local search. One of the goals of a *hyper-heuristic* is to provide solution methodologies which are more general than currently possible [9].

Two types of hyper-heuristics are distinguished in the literature, namely constructive and improvement based hyper-heuristics [9]. Constructive based hyper-heuristics start with an empty timetable, and select low level heuristics to build a solution step by step. Improvement based hyper-heuristics start with an initial solution and, at each iteration, selects appropriate improvement low level heuristics to perturb the solution. These two types of approaches can be further extended to on-line or off-line approaches, based on the learning methods employed. In on-line hyper-heuristics, the learning takes place during the problem solving. In off-line hyper-heuristics, the learning happens during the training phase before solving other problem instances (see [10]). Our work concentrates on on-line constructive hyper-heuristics for solving examination timetabling problems.

Some hyper-heuristic approaches have been studied for examination timetabling problems. These include tabu search [5, 7, 19, 20], case-based reasoning [8, 41], variable neighborhood search [6, 34, 35], graph based methods [2, 5, 7, 34, 35], memetic algorithms [16], heuristic combinations [31] and genetic programming [32, 33]. More details of these hyper-heuristics can be found in a recent survey by Burke et al. [6].

In [7], a tabu search was employed to hybridize six constructive graph coloring heuristics to solve both exam and course timetabling problems. Each move of the tabu search generates a new heuristic list by randomly changing some heuristics in the previous list. The low level heuristics in the list are sequentially applied to select exams, one by one, in order to construct a feasible timetable. In [5], knowledge discovery techniques were used within a case based reasoning system, to recommend graph coloring heuristics in constructing exam timetables. A tabu search approach in [7] explores the search space of two constructive graph coloring heuristics, saturation degree and largest degree, and was shown to outperform a systematic heuristic hybridization approach and a case based reasoning system on a set of random and four of the Carter benchmark instances in examination timetabling. In both of these works, the high level search (tabu search, case based reasoning and systematic hybridization) explores the search space of heuristics.

Kendall and Hussin [19] developed a tabu search based hyper-heuristic. The process begins by generating an initial solution based on saturation degree or largest degree. This initial solution may not be feasible (i.e. not all exams can be scheduled). The neighborhood of the initial solution is then examined in order to improve the generated timetable. Four low-level heuristics (move an exam, swap exams, select and schedule an exam and remove an exam) are utilized within a tabu search framework. The heuristic that produces the best improvement is applied next. The process is repeated until a time limit is reached or there are no improvements for a predefined number of iterations.

Qu and Burke [34] studied the effectiveness of using different high-level search algorithms including steepest descent, tabu search, iterated local search and variable neighborhood search in a graph based hyper-heuristic framework for examination timetabling. Experimental results showed that the choices of particular neighborhoods for the high level algorithms are not crucial within the hyper-heuristic framework, at least for the Carter benchmark datasets. An analysis on the characteristics of the two search spaces showed that the high level search is able to "jump" within the solution space thus exploring a larger area of the space. Good results obtained for both exam and course timetabling problems are comparable to state-of-the-art approaches. The difference between Qu and Burke [34] and our work is that, in our approach, the low level heuristics are hybridized hierarchically and are simultaneously applied to identify the most difficult exams based on a *difficulty index;* whilst, in [34], the low level heuristics are applied sequentially.

Pillay and Banzhaf [31] also studied hierarchical heuristic combinations in a hyper-heuristic framework for the examination timetabling problem, where heuristics in each combination are implemented simultaneously rather than sequentially. A set of low level heuristics are used to construct heuristic combinations (largest degree, largest enrollment, largest weighted degree, saturation degree, and a new low-level heuristic called highest cost). Four heuristic combinations are generated using two or three low level heuristics. Exams are ordered based on heuristic combinations and the most difficult exam is scheduled in the timeslot causing the minimum penalty. Experimental results on the Carter benchmark datasets are comparable to those obtained by other hyper-heuristic approaches. Our work is similar to [31] in that, we also use four lists which are generated by the hybridization of low level heuristics. However, the difference in our work is that the four lists are generated by the hybridization of four low level heuristics, whilst in [31], each list contains the combination of two or three heuristics. Exams are scheduled based on heuristic combinations to the minimum penalty timeslot, whilst in our work, exams are selected based on their *difficulty index* and are scheduled in an appropriate timeslot, which is selected by a roulette wheel selection mechanism.

Motivated by the above works, our approach utilizes a *difficulty index*, which is a mechanism to select the most difficult exams based on hierarchical heuristic hybridizations. A probabilistic timeslot selection method is also developed to further improve the efficiency of the approach.

3 Related work on ITC (International Timetabling Competition) 2007 examination timetabling track

Based on the success of the first International Timetabling competition in 2003, organized by the Metaheuristic Network, the second International Timetabling Competition (ITC 2007) was released. The ITC 2007 includes one track on examination timetabling and two tracks on course timetabling (post enrollment based course timetabling and curriculum based course timetabling). The aim of the competition is to facilitate a better understanding of real world timetabling problems and to reduce the gap between research and practice [23, 25, 26]. There have been a number of papers tackling the problem instances of the examination track at ITC 2007.

Müller [29] developed a three phase constraint-based solver. The first phase uses an Iterative Forward Search algorithm to construct an initial feasible solution. At each iteration, an exam is selected and allocated to a room and timeslot. Backtracking is employed to unschedule exams that are already in the schedule and cause a conflict with the exam being scheduled. In the second phase, hill climbing is carried out where a list of neighborhoods to be applied including swapping or changing timeslots and rooms for randomly selected exams is determined based on a probability. Hill climbing stops after a specified number of iterations or when no further improvement is possible. Finally a great deluge

algorithm is applied to improve the solution from the second phase. Müller was the competition winner of the ITC 2007 examination track and achieved the best results for ten of the twelve instances.

Gogos et al. [17, 18] applied a GRASP based heuristic to the ITC 2007 datasets. In the construction phase, initial solutions are generated based on five lists of exams, constructed by using different criteria. A tournament based method is then used to choose the exam to be scheduled until all lists are empty. A backtracking strategy, employing a tabu list, is also used. In the improvement phase, a simulated annealing procedure improves the initial solutions. In the third phase, branch and bound is used to analyze timeslots that need room changes. Timeslots are then selected based on the overall satisfaction of particular soft constraints. Based on the ITC 2007 evaluation, Gogos was ranked second in the competition.

Atsuta et al. [3] applied a general purpose solver, combining iterated local search and tabu search. The solver begins with predetermined initial weights and then the differences between the weights of soft and hard constraints are dynamically controlled during the process. The instances are represented using linear 0–1 inequalities and quadratic 0–1 inequalities with all-different constraints. The technique is very effective across all tracks of the competition and was placed third in the examination timetabling track.

De Smet [14] incorporated tabu search techniques within the Drools Solver, an open-source business rule management system. Constraints are written in the drools rule language. Exams are ordered based on their size and duration and scheduled into the 'best' timeslot chosen by a placement heuristic. The tabu search employs three neighborhoods (timeslot change, room change and exam swap). The Drools Solver was placed fourth in the examination timetabling track.

Pillay [30] developed an approach based on cell biology. Exams are ordered using saturation degree and sequentially scheduled to available "cells" (timeslots) that cause the minimum penalty with ties being randomly broken. Rooms are selected using the best fit heuristic. If no feasible cells remain, the exams which have already been placed are moved to a cell causing the minimal overall soft constraint penalty (called *cell division*). If this is not possible, *cell interaction*, employing a swapping operation, will be employed to remove the hard constraint violations. This process is repeated to find a feasible solution, which is then improved by *cell migration*, by heuristically swapping the contents of cells with equal durations. Pillay [30] was placed fifth in the ITC 2007 examination track.

McCollum et al. [25] employed an extended great deluge technique that was originally proposed by McMullan [27] to solve ITC 2007 examination timetabling problems. The extended great deluge technique employs a reheating mechanism which is invoked after a predefined number of consecutive non improvement moves to recalculate the initial level and decay rate. The author carried out a large number of runs (e.g. 51 runs for each instance) compared to the other approaches in the ITC 2007 competitions (e.g. 11 runs for each instance). The proposed approach outperformed others on six of the ITC 2007 datasets.

4 A Graph Coloring Constructive Hyper-Heuristic (GCCHH) algorithm

Burke et al. [7], Ayob et al. [4] and Qu et al. [36] argued that although simple graph heuristics have been widely used, a single strategy in determining the difficulty level of the exam to be scheduled in solution construction is not sufficient to provide high quality solutions. Therefore, in our proposed GCCHH approach, the difficulty level of scheduling an exam is determined by hybridizing several graph coloring heuristics simultaneously. In GCCHH, at the higher level, the *difficulty index* of exams is calculated by using hierarchical hybridisations of four graph coloring heuristics. The most difficult exam to be scheduled (i.e. the one with minimum difficulty index) will be allocated first to the appropriate timeslot determined by a roulette wheel selection mechanism. If two or more exams have the same *difficulty index*, one exam will be selected randomly. At the lower level, four low level graph coloring heuristics are used to generate four heuristic hybridisations. Algorithm 1 shows the pseudo-code of the GCCHH approach.

In our work, we use four graph coloring heuristics (other heuristics might also be applicable). These heuristics have been chosen because they are the most widely applied graph coloring heuristics and have produced good quality solutions for examination timetabling problems [2, 5, 7, 19]. The heuristics we use are:

- Largest Degree First (LD): exams are ordered, in decreasing order, by the number of conflicts they have with all other exams.
- Saturation Degree First (SD): exams are ordered dynamically, in ascending order, by the number of remaining timeslots.
- Largest Colored Degree First (LCD): exams are ordered based on LD, but only considering the number of conflicts with those already scheduled exams in non-increasing order.
- Largest Enrolment First (LE): exams are ordered by the number of students enrolled, in decreasing order.

In this work, we use four heuristic hybridizations to order exams which produce four sorted lists (see Table 1).

Algorithm 1 Pseudo-code of the GCCHH approach

1. Initialization
   Generate the four ordering lists of exams by using h_i, $i = 1, \ldots, 4$ (Table 1 and Algorithm 2)
2. While (ordering lists are not empty) do
   {
   a. Calculate the *difficulty index* of the first exam in each list using (1).
   b. Select the exam e that has the minimum *difficulty index*. Randomly select one exam if there is more than one exam with the same difficulty index.
   c. Assign exam e to the timeslot selected by Roulette_wheel_timeslots_selection (Algorithm 3)
   d. Remove e from all ordering lists.
   e. Reorder all ordering lists using h_i.
   }
   Verify if all the hard constraints are satisfied.
   If (ordering lists are empty and all hard constraints are satisfied) then return the feasible solution.
   Otherwise, return infeasible solution.

Table 1 The four heuristic hybridization strategies

h_i	Heuristic hybridizations
h_1	In this heuristic hybridization ($LD + LE + SD + LCD$), the exams to be scheduled are arranged in a non-increasing order of the number of conflicts they have with other exams (*LD*); those with equal *LD* evaluations are then arranged in a non increasing order of the number of student enrollments (*LE*), then in a non decreasing order of the number of available timeslots (*SD*) and, finally, in a non-increasing order of the number of conflicts the exam has with those already scheduled (*LCD*).
h_2	Similar to h_1, this heuristic hybridization ($SD + LCD + LD + LE$) arranges the exams to be scheduled by using *SD*, *LCD*, *LD* and *LE* hierarchically.
h_3	Same as the above, with a different hierarchy of heuristics ($LCD + SD + LD + LE$).
h_4	Same as the above, with a different hierarchy of heuristics ($LE + LD + SD + LCD$).

Algorithm 2 presents the pseudo-code of h_1 which constructs a list of exams ordered by using the heuristic hybridization ($LD + LE + SD + LCD$). Similarly the other three lists of exams are constructed by using h_2, h_3 and h_4, respectively. In this work, we define dynamic or static lists based on the first ranking criteria. We then have two dynamic lists (i.e. h_2 and h_3, which order exams dynamically by the number of remaining timeslots or by the number of conflicts with those exams already assigned) and two static lists (i.e. h_1 and h_4). The idea of hybridizing different heuristics (dynamic and static) is motivated by their different performances during the solution construction and for different problem instances. Some of heuristics may work well at the beginning (especially *LE* and *LD*), whilst others may work well at the end of solution construction. For example, saturation degree (*SD*) may not be efficient at the beginning since most timeslots are not occupied. Whereas largest degree (*LD*) may be efficient at the beginning since it selects the most conflicted exam to be scheduled first [34]. In our GCCHH approach, we employ four different heuristics (either dynamic or static) to overcome the shortcoming of using single heuristics.

Algorithm 2 Pseudo-code of heuristic hybridization *h1*

***h1* (exam e1, exam e2)**
Begin

```
If (e1.LD = e2.LD and e1.LE ≠ e2.LE)
     if(e1.LE > e2.LE)
     Return e1 > e2
     Else Return e2 > e1
Else if (e1.LD = e2.LD
and e1.LE = e2.LE and
e1.SD ≠ e2.SD)
     if(e1.SD < e2.SD)
     Return e1 < e2
     Else Return e2 < e1
Else if (e1.LD = e2.LD and
e1.LE = e2.LE and e1.SD = e2.SD
and e1.LCD ≠ e2.LCD)
     If(e1.LCD > e2.LCD)
     Return e1 > e2
     Else Return e2 > e1
Else return e1 > e2
```

End

h_1, h_2, h_3 and h_4 serve as the low level heuristics for the GCCHH approach (see Algorithm 1). A low level heuristic prioritizes exams according to the level of difficulty in scheduling them into the timetable. The rationale behind this is to make sure that the most difficult exams are scheduled first [7].

Based on the four ordered lists of exams constructed by using h_1–h_4, we introduce the difficulty index which is calculated for the first exam of each ordered list using (1).

$$F(e_i) = \sum_{k=1}^{4} I_{ki} \tag{1}$$

where I_{ki} is the order of exam e_i in the list produced by heuristic h_k.

By using the difficulty index, the first exam across all lists which has the minimum difficulty index, $F(e_i)$, will be chosen to be scheduled at each iteration of the solution construction.

Table 2 presents an illustrative example of calculating the *difficulty index*. Assume by using h_1–h_4, exams to be sched-

Table 2 An illustrative example of exam selection by using difficulty index

Heuristics	Order						
	1	2	3	4	5	6	7
$LD+LE+SD+LCD$	e_3	e_5	e_1	e_6	e_7	e_4	e_2
$SD+LCD+LD+LE$	e_7	e_5	e_6	e_3	e_2	e_4	e_1
$LCD+SD+LD+LE$	e_6	e_3	e_5	e_7	e_4	e_1	e_2
$LD+LE+SD+LCD$	e_4	e_2	e_3	e_7	e_1	e_5	e_6

uled are ordered as shown in Table 2. By using (1), the difficulty indexes for the first exam in each list are as follows:

- $e_3 = 1+4+2+3 = 10$
- $e_7 = 5+1+4+4 = 14$
- $e_6 = 4+3+1+7 = 15$
- $e_4 = 6+6+5+1 = 18$

Since exam e_3 has the minimum difficulty index, it will be scheduled at this iteration.

By using this mechanism, the difficulty level to schedule an exam is determined based on multiple criteria rather than a single ordering strategy.

In constructing an exam timetable, heuristics on exam selection have been most widely studied, utilizing a range of ordering criteria. However, timeslot selection for the chosen exam is not so often considered. Usually, the timeslot which causes the minimum penalty is selected, ties are broken by choosing the first timeslot [7, 34], or one at random [7, 34]. This may not necessarily guarantee a good quality, or even feasible solution in some cases. Indeed, they always produce the same solutions (except for a random approach) for every run. Therefore, in this work, we propose a probabilistic timeslot selection mechanism, roulette wheel selection, to choose the appropriate timeslot so that we (usually) are returned a different solution from every run. Moreover, this selection mechanism may potentially produce good quality solutions since timeslots which cause smaller penalties are more likely to be chosen compared to those with a higher penalty. Algorithm 3 presents the pseudo-code of the roulette wheel selection method.

In roulette wheel timeslot selection, the segment span $S(i)$ for each feasible timeslot is determined based on the penalties of assigning the chosen exam by using (2) as follows:

$$S(i) = S(i-1) + (1 - C(i)) \Big/ \sum_{k=1}^{T} C(k) \tag{2}$$

where i is the ith feasible timeslot, $S(i)$ is the segment span of the ith timeslot and $C(i)$ is the cost for the ith timeslot and $S(0) = 0$.

By using roulette wheel selection a timeslot is chosen in proportion to the segment span calculated based on the sum of penalties.

Algorithm 3 Pseudo-code of Roulette wheel selection for timeslot selection

Roulette_ wheel_timeslots_selection (**exam** e)
Begin

- Find all feasible timeslots for exam e
- Calculate the cost for each feasible timeslot $C(i)$
- Calculate the segment span for each available timeslots $S(i)$ using (2)
- Generate a random number r between [0, 1]
- Select timeslot s where r falls in its segment span
- Assign exam e to timeslot s
- Return true

End

Table 3 Characteristics of Carter's un-capacitated examination timetabling benchmark dataset [13]

Data sets	Number of timeslots	Number of exams	Number of students
Car-f-92-I	32	543	18419
Car-s-91-I	35	682	16925
Ear-f-83-I	24	190	1125
Hec-s-92-I	18	81	2823
Kfu-s-93	20	461	5349
Lse-f-91	18	381	2726
Pur-s-93-I	43	2419	30032
Rye-s-93	23	486	11483
Sta-f-83-I	13	139	611
Tre-s-92	23	261	4360
Uta-s-92-I	35	622	21267
Ute-s-92	10	184	2750
Yor-f-83-I	21	181	941

5 Experimental results

We evaluate our GCCHH approach on two sets of benchmark examination timetabling datasets. These are Carter's un-capacitated (Toronto b Type I in [36]) and the ITC 2007 [23, 25, 26] examination timetabling benchmark datasets. The algorithm was developed using Microsoft Visual C++ 6.0 and all simulations were run on a Windows XP 2002 machine with an AMD Athlon 1.92 GHz processor with 512 MB of RAM.

5.1 GCCHH for Carter's uncapacitated benchmark dataset

The proposed GCCHH was evaluated on Carter's un-capacitated examination timetabling benchmark dataset which contains 13 instances. It was introduced by Carter et al. [13] and extended to several variants over the years. We test our GCCHH approach on the Toronto b variant [36].

Table 4 Results of GCCHH compared against the best results of current graph coloring based hyper-heuristics in the literature

Datasets	Our results		FZLO	VNS	HGH	TS	GHH	HC	NON-HH
	Best	Average							
Car-f-92-I	4.70	4.93	4.52	4.7	–	4.53	**4.16**	4.28	3.93
Car-s-91-I	5.14	5.26	5.2	5.4	–	5.36	5.16	**4.97**	4.5
Ear-f-83-I	37.86	38.77	37.02	37.29	45.60	37.92	**35.86**	36.86	29.3
Hec-s-92-I	11.90	12.01	**11.78**	12.23	–	12.25	11.94	11.85	9.2
Kfu-s-93	15.30	15.94	15.81	15.11	–	15.2	14.79	**14.62**	13.0
Lse-f-91	12.33	13.10	12.09	12.71	–	11.33	**11.15**	11.14	9.6
Pur-s-93-I	5.37	6.76	–	–	–	–	–	**4.37**	–
Rye-s-93	10.71	11.06	10.35	–	–	–	–	**9.65**	6.8
Sta-f-83-I	160.12	161.56	160.42	158.8	158.2	**158.19**	159.00	158.33	157.2
Tre-s-92	**8.32**	9.01	8.67	8.67	–	8.92	8.6	8.48	7.9
Uta-s-92-I	3.88	4.13	3.57	3.54	4.52	3.88	3.59	**3.4**	3.14
Ute-s-92	32.67	33.73	**27.78**	29.68	35.40	28.01	28.3	28.88	24.4
Yor-f-83-I	**40.53**	41.84	40.66	43.0	–	41.37	41.81	40.74	36.2

Note:
FZLO: Fuzzy multiple graph coloring ordering criteria proposed by Asmuni et al. [2]
VNS: Hybrid variable neighbourhood hyper-heuristics proposed by Qu and Burke [35]
HGH: Hybrid graph heuristics in hyper-heuristics proposed by Burke et al. [5]
TS: A graph based hyper-heuristic proposed by Burke et al. [7]
GHH: Hybridisations within a graph based hyper-heuristic framework proposed by Qu and Burke [34]
HC: A study of heuristic combinations for hyper-heuristic systems proposed by Pillay and Banzhaf [31]
NON-HH: Best reported results (non hyper-heuristics) Qu et al. [36]. Note that all compared methods are single pass methods except TS, GHH and NON-HH methods

Table 3 presents the characteristics of this dataset. Twenty independent runs (each set of 20 run taking 1–4 hours depending on the size of the problem instance) were carried out for each of the 13 instances using different random seeds. Please note that this run time is acceptable in university timetabling problems because the timetables are usually produced months before the actual schedule is required [36].

Table 4 lists the best and average results of GCCHH for each of the benchmark instances. The results represent the calculation of soft constraints violations. They are comparable to the current state-of-the-art hyper-heuristics employing graph coloring heuristics (best results from the literature using hyper-heuristic approaches are shown in bold). Compared to the other hyper-heuristics, GCCHH obtained the best results on two out of 13 instances. We also provide the best results obtained by non-hyper-heuristic approaches (bespoke methods) in the literature in the last column of Table 4. Note that our GCCHH approach is a single pass stochastic method, rather than being an iterative process (i.e. it is just a constructive heuristic). We are not aiming to beat these bespoke methods which usually employ both a constructive phase and an improvement phase in obtaining the best results.

As shown in Table 4, GCCHH obtained comparable results across all methods for all tested instances. It is able to obtain better solutions on 5 (Kfu-s-93, Pur-s-93, Sta-f-83, Tre-s-92 and Yor-f-83) and 7 (Car-s-91, Hec-s-92, Lse-f-91, Pur-s-93, Rye-s-93, Tre-s-92 and Yor-f-83) instances compared to FZLO and VNS, respectively. On the other hand, when compared to HGH and TS, GCCHH is able to achieve better results on 3 (Ear-f-83, Uta-s-92 and Ute-s-92) and 4 (Ear-f-83, Hec-s-92, Tre-s-92 and Yor-f-83) instances, respectively. Note that, HGH was tested on 4 out of 13 instances and the best results obtained from graph coloring hyper-heuristics in TS are further improved by applying a steepest decent local search. Our approach also outperformed GHH and HC on 4 (Car-s-91, Hec-s-92, Tre-s-92 and Yor-f-83) and 2 (Tre-s-92 and Yor-f-83) instances, respectively. Furthermore, only HC reported results for Pur-s-93 and Rye-s-93 instances. Whilst, GHH, TS, HGH and VNS did not report results on Rye-s-93 and Pur-s-93 datasets (we suspect, due to the inconsistence of the data files for these instances). Moreover, all the compared methods have employed complex mechanisms such as fuzzy logic, tabu search or variable neighbourhood to select the most suitable low level heuristics. Whilst, in GCCHH, we only use a simple method to chose the low level heuristic and produce competitive results compared to other methods.

5.2 GCCHH for the ITC 2007 dataset

To apply GCCHH to the ITC 2007 datasets [23, 25, 26], we have made small changes to the low level heuristics. Due to

Table 5 The ITC 2007 benchmark examination timetabling datasets

Data sets	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
Dataset 1	7891	607	54	7	5	7	5	10	100	30	5	7833
Dataset 2	12743	870	40	49	5	15	1	25	250	30	5	12484
Dataset 3	16439	934	36	48	10	15	4	20	200	20	10	16365
Dataset 4	5045	273	21	1	5	9	2	10	50	10	5	4421
Dataset 5	9253	1018	42	3	15	40	5	0	250	30	10	8719
Dataset 6	7909	242	16	8	5	20	20	25	25	30	15	7909
Dataset 7	14676	1096	80	15	5	25	10	15	250	30	10	13795
Dataset 8	7718	598	80	8	0	150	15	25	250	30	5	7718

Note:
A1: No. of students reported in [23]
A2: Number of exams
A3: Number of timeslots
A4: Number of rooms
A5: Two in a day penalty
A6: Two in a row penalty
A7: Timeslots spread penalty
A8: No mixed duration penalty
A9: Number of largest exams
A10: Number of last timeslots to avoid
A11: Front load penalty
A12: Number of actual students in the datasets

the size of the dataset in ITC 2007, we reduced the number of heuristic hybridizations. That is, for each low-level heuristic, we only hybridize two graph coloring heuristics as follows:

1. $h_1(LD + LE)$: exams are ordered decreasingly by the number of conflicts and then those with the same number of conflicts ordered decreasingly by student enrollments.
2. $h_2(SD + LCD)$: similar to the above but using SD and LCD for ordering.
3. $h_3(LCD + SD)$: Same as above, with a slightly different ordering.
4. $h_4(LE + LD)$: Same as above, with a slightly different ordering.

To reduce the computational time, each exam is randomly allocated to the best feasible timeslot and a best fit room. Two repair mechanisms (timeslot and room repair mechanisms) are used to repair violations of the additional hard constraints present in the ITC 2007 dataset.

- In the timeslot repair mechanism, we verify all timeslot related hard constraints by moving or swapping appropriate exams. For example, if Exam_A must be scheduled after Exam_B, then the timeslot repair mechanism will be applied to satisfy this constraint.
- In the room repair mechanism, we verify all room related hard constraints by moving all exams violating this constraint. For example, if Exam_A cannot share a room with other exams, then all exams assigned to the same room as Exam_A will be moved to other suitable rooms.

Both the timeslot and room repair mechanisms are invoked when all exams have been scheduled. The algorithm terminates after a given time and returns either a feasible, if the repair process is successful, or an infeasible solution if it is not successful.

Computational experiments were carried out on eight (actually there are 12 instances listed on the web site but only eight was available at the time of experimentation) instances from the ITC 2007 examination timetabling track. Details of problem descriptions and specification can be found in [23, 25, 26]. Table 5 shows the main characteristics of these instances.

The GCCHH approach was run 20 times on each instance with different random seeds. The stopping condition is set at 650 seconds (determined by benchmark programs provided at the ITC 2007 web site). Table 6 presents our experimental results (best and average) which represent the calculation of soft constraints violations. The best results in the literature are shown in bold. The results show that GCCHH is competitive with the ITC 2007 competition winners' approaches, and in two cases (Datasets 2 and 4), outperformed all other approaches from across the literature (including bespoke approaches).

Note that [3] and [30] are single pass methods. It is not really fair to compare our results with those of [18, 29] and [14], as their algorithms employed two-phase approaches where a constructive phase is followed by an improvement phase (as explained in Sect. 3). Whilst, our method only employs a constructive phase. Our approach was outperformed

Table 6 Results of GCCHH on the ITC 2007 compared to the best results of ITC 2007 winners

Data sets	Our results		Müller [29]	Gogos et al. [18]	Atsuta et al. [3]	De Smet [14]	Pillay [30]
	Best	Average					
Dataset 1	6234	6311	**4370**	5905	8006	6670	12035
Dataset 2	**395**	400	400	1008	3470	623	3074
Dataset 3	13002	13120	**10049**	13862	18622	–	15917
Dataset 4	**17940**	18011	18141	18674	22559	–	23582
Dataset 5	3900	3986	**2988**	4139	4714	3847	6860
Dataset 6	27000	27420	**26950**	27640	29155	27815	32250
Dataset 7	6214	6345	**4213**	6683	10473	5420	17666
Dataset 8	8552	8624	**7861**	10521	14317	–	16184

Table 7 Results of GCCHH on the ITC 2007 examination timetabling datasets compared to the best results of the Post-ITC 2007 approaches

Data sets	Our results		Gogos et al. [18]	Post-Müller	McCollum et al. [25]
	Best	Average			
Dataset 1	6234	6311	4775	**4370**	4633
Dataset 2	395	400	**385**	**385**	405
Dataset 3	13002	13120	**8996**	9378	9064
Dataset 4	17940	18011	16204	**15368**	15663
Dataset 5	3900	3986	**2929**	2988	3042
Dataset 6	27000	27420	**25740**	26365	25880
Dataset 7	6214	6345	4087	4138	**4037**
Dataset 8	8552	8624	7777	7516	**7461**

[29] on 2 instances, and outperformed [18] and [14] on 7 and 3 datasets, respectively. Note that the algorithm by [14] did not obtain a feasible solution for three datasets. Indeed, when compared against constructive methods [3, 30], our method obtained better results on all tested instances. Again, we can see that the results obtained by GCCHH are comparable with other methods (or even better in some cases). This indicates that GCCHH is a simple yet efficient method which outperformed existing constructive methods and is competitive with the best bespoke two-phase based methods on the ITC 2007 datasets.

Table 7 shows the comparison of our results with the post-ITC 2007 approaches. Best results are shown in bold.

Note that the post-ITC 2007 approaches performed more runs compared to the competition winners, i.e. [25] performed 51 runs, Post-Müller performed 100 runs and [17] performed 100 runs, for each instance, see more details in [25].

As can be seen from Table 7, our results are competitive when compared to all post-ITC 2007 approaches. It outperformed the algorithm in [25] on one dataset. Again, our results are compared with those post-ITC 2007 approaches where two phases are employed.

6 Discussion

As shown in Sects. 5.1 and 5.2, in both set of experiments (Carter and ITC 2007 datasets), GCCHH obtained competitive results when compared against existing best methods in the literature.

For the Carter dataset, all compared hyper-heuristic methods employed single sequential graph coloring heuristics with an intelligent high level strategy to determine the most suitable heuristics (heuristics lists). The results obtained by our GCCHH reveal that hierarchically combining graph coloring heuristics using difficulty index mechanisms is able to produce competitive results.

For the ITC 2007 datasets, all compared methods are specifically designed to solve the ITC 2007 problems. In [3], the constructive methods used constraint satisfaction problems solver; whilst the other constructive method [30] used a biological approach which mimics cell behavior. The results show that our GCCHH approach is able to produce competitive results compared to these constructive heuristics (with regard to ITC 2007 datasets). In [14], the initial solution generated by a single graph coloring heuristic is improved by tabu search. The approach obtained feasible timetables for 5 out of 8 datasets. This indicates that sin-

gle pass graph coloring heuristics are not efficient on their own or even used together with meta-heuristic algorithms. However, when hybridizing several graph coloring heuristics, the combinations of graph coloring heuristics are able to produce feasible timetables for all tested instances without any improvements.

Given that our method is a construction phase only approach the results produced are very competitive indeed. Furthermore the approach is simple enough to be easily implemented. It can be seen that in both benchmark examination timetabling problems, our constructive approach which is a single pass stochastic method is able to produce competitive results compared with current best approaches in the literature. We believe this is due to the use of a *difficulty index* to select the most difficult exams based on the simultaneous use of dynamic and static heuristics lists and the use of the roulette wheel selection for timeslot assignment. Furthermore, dynamic and static heuristics, when hierarchically hybridized, is able to select the most difficult exams to be scheduled first based on different ordering criteria. Moreover, there are only two parameters that need to be tuned in our GCCHH which are the length of the hybrid heuristic lists and the stopping condition. Whilst, most of compared methods have some parameters that need to be tuned in advance in order to obtain the best results.

Overall, our approach does not rely on complicated search approaches to find out how to sequentially employ low level heuristics. Rather it provides a general mechanism to construct the solution. It is simple to implement, and can be easily hybridized with any other meta-heuristics. In another words, it provides a fundamentally simple constructive approach which is general enough.

7 Conclusion

This work has proposed a new hybrid constructive hyper-heuristic for solving examination timetabling problems. Widely used graph coloring heuristics are hybridized to generate four new low level heuristics via the construction of ordered lists. A difficulty index is used to determine which exam is to be scheduled first based on the order of the exam across all lists. The most difficult exam to be scheduled is scheduled first (i.e. the one with the minimum difficulty index). The index is calculated based on a combined, rather than a single, ordering criteria. Results of the approach on two sets of benchmark examination timetabling datasets (Carter and ITC 2007) show that this simple, yet efficient, approach produces competitive or better results when compared to the best approaches in the literature (with regards to Carter and ITC 2007 datasets).

References

1. Abdullah S, Turabieh H, McCollum B (2009) A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems. In: Proceedings of the 6th international workshop on hybrid metaheuristics. Lecture notes in computer science, vol 5818. Springer, Berlin, pp 60–72
2. Asmuni H, Burke EK, Garibaldi J, McCollum B (2005) Fuzzy multiple ordering criteria for examination timetabling. In: Burke EK, Trick M (eds) Practice and theory of automated timetabling V: selected papers from the 5th international conference. Lecture notes in computer science, vol 3616. Springer, Berlin, pp 334–353
3. Atsuta M, Nonobe K, Ibaraki T (2007) ITC2007 Track 1: an approach using general CSP solver. www.cs.qub.ac.uk/itc2007
4. Ayob M, Malik AMA, Abdullah S, Hamdan AR, Kendall G, Qu R (2007) Solving a practical examination timetabling problem: a case study. In: Gervasi O, Gavrilova M (eds) ICCSA 2007, Part III. LNCS, vol 4707. Springer, Heidelberg, pp 611–624
5. Burke EK, Dror M, Petrovic S, Qu R (2005) Hybrid graph heuristics in hyper-heuristics applied to exam timetabling problems. In: Golden BL, Raghavan S, Wasil EA (eds) The next wave in computing, optimisation, and decision technologies. Springer, Maryland, pp 79–91
6. Burke EK, Eckersley AJ, McCollum B, Petrovic S, Qu R (2010) Hybrid variable neighbourhood approaches to university exam timetabling. Eur J Oper Res 206:46–53
7. Burke EK, McCollum B, Meisels A, Petrovic S, Qu R (2007) A graph based hyper-heuristic for exam timetabling problems. Eur J Oper Res 176:177–192
8. Burke EK, Petrovic S, Qu R (2006) Case-based heuristic selection for timetabling problems. J Sched 9:115–132
9. Burke EK, Hyde M, Kendall G, Ochoa G, Ozcan E, Woodward J (2009a) A classification of hyper-heuristics approaches. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, international series in operations research & management science. Springer, Berlin
10. Burke EK, Hyde M, Kendall G, Ochoa G, Ozcan E, Woodward J (2009b) Exploring hyper-heuristic methodologies with genetic programming, computational intelligence: collaboration, fusion and emergence. In: Mumford C, Jain L (eds) Intelligent systems reference library. Springer, Berlin, pp 177–201
11. Burke E, Hart E, Kendall G, Newall J, Ross P, Schulenburg S (2003) Hyperheuristics: an emerging direction in modern research technology. In: Handbook of metaheuristics. Kluwer Academic, Dordrecht, pp 457–474 (Chap 16)
12. Burke EK, Bykov Y, Newall JP, Petrovic S (2004) A time-predefined local search approach to exam timetabling problems. IIE Trans Oper Eng 36(6):509–528
13. Carter MW, Laporte G, Lee SY (1996) Examination timetabling: algorithmic strategies and applications. J Oper Res Soc 47(3):373–383
14. De Smet G (2008) ITC2007—examination track, practice and theory of automated timetabling (PATAT 2008), Montreal, 19–22 August 2008
15. Eley M (2007) Ant algorithms for the exam timetabling problem. In: Burke EK, Rudova H (eds) PATAT 2007. LNCS, vol 3867. Springer, Heidelberg, pp 364–382
16. Ersoy E, Özcan E, Etaner AS (2007) Memetic algorithms and hyperhill-climbers. In: Proceedings of the 3rd multidisciplinary international conference on scheduling: theory and applications, Paris, France, August 2007, pp 159–166
17. Gogos C, Alefragis P, Housos E (2010) An improved multi-staged algorithmic process for the solution of the examination timetabling problem. Ann Oper Res. doi:10.1007/s10479-010-0712-3

18. 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
19. Kendall G, Hussin NM (2005a) An investigation of a tabu search based hyper-heuristic for examination timetabling. In: Kendall G, Burke E, Petrovic S (eds) Selected papers from multidisciplinary scheduling; theory and applications, pp 309–328
20. Kendall G, Hussin NM (2005b) A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA University of Technology. In: Burke EK, Trick M (eds) Practice and theory of automated timetabling V: selected papers from the 5th international conference. Lecture notes in computer science, vol 3616. Springer, Berlin, pp 199–218
21. Li J, Burke EK, Qu R (2011) A pattern recognition based intelligent search method and two assignment problem case studies. Appl Intell. doi:10.1007/s10489-010-0270-z
22. Mansour N, Isahakian V, Ghalayini I (2011) Scatter search technique for exam timetabling. Appl Intell 34:299–310
23. McCollum B, McMullan P, Burke EK, Parkes AJ, Qu R (2007) A new model for automated examination timetabling. Submitted post PATAT08 special issue of J. Sched. Available as technical report QUB/IEEE/Tech/ITC2007/Exam/v4.0/17 from http://www.cs.qub.ac.uk/itc2007/examtrack/exam_track_index.htm
24. McCollum B (2007) A perspective on bridging the gap between research and practice in university timetabling. In: Burke EK, Rudova H (eds) Practice and theory of automated timetabling VI. Lecture note in computer science, vol 3867. Springer, Berlin, pp 3–23
25. McCollum B, McMullan P, Parkes A, Burke E, Abdullah S (2009) An extended great deluge approach to the examination timetabling problem. In: Proceedings of MISTA09. The 4th multidisciplinary international conference on scheduling: theory and applications, Dublin, August 2009, pp 424–434
26. McCollum B, McMullan P, Paechter B, Lewis R, Schaerf A, Di Gaspero L, Parkes AJ, Qu R, Burke E (2010) Setting the research agenda in automated timetabling: the second international timetabling competition. INFORMS J Comput 22(1):120–130
27. McMullan P (2007) An extended implementation of the great deluge algorithm for course timetabling. In: Lecture notes in computer science, vol 4487. Springer, Berlin, pp 538–545
28. Merlot LTG, Boland N, Hughes BD, Stuckey PJ (2003) A hybrid algorithm for the examination timetabling problem. In: Burke EK, De Causmaecker P (eds) Practice and theory of automated timetabling: selected papers from the 4th international conference. Lecture notes in computer science, vol 2740. Springer, Berlin, pp 207–231
29. Müller T (2009) ITC2007 solver description: a hybrid approach. Ann Oper Res 172(1):429–44
30. Pillay A (2007) Developmental approach to the examination timetabling problem. www.cs.qub.ac.uk/itc2007
31. Pillay N, Banzhaf W (2009) A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. Eur J Oper Res 197(2):482–491
32. Pillay N, Banzhaf W (2007) A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In: Neves A et al (eds) Progress in artificial intelligence. Lecture notes in artificial intelligence, vol 4874. Springer, Berlin, pp 223–234
33. Pillay N (2008) An analysis of representations for hyper-heuristics for the uncapacitated examination timetabling problem in a genetic programming system. In: Proceeding of SAICSIT 2008. ACM Press, New York, pp 188–192
34. Qu R, Burke EK (2009) Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. J Oper Res Soc 60:1273–1285
35. Qu R, Burke EK (2005) Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems. In: Proceedings of the MIC2005: the sixth metaheuristics international conference, Vienna, Austria, August 2005
36. Qu R, Burke EK, McCollum B, Merlot LTG, Lee SY (2009) A survey of search methodologies and automated system development for examination timetabling. J Sched 12(1):55–89
37. Ross P, Hart E, Corne D (1998) Some observations about GA-based exam timetabling. In: Burke EK, Carter MW (eds) Practice and theory of automated timetabling: selected papers from the 2nd international conference. Lecture notes in computer science, vol 1408. Springer, Berlin, pp 115–129
38. Sabar NR, Ayob M, Kendall G (2009b) Tabu exponential Monte-Carlo with counter heuristic for examination timetabling. In: Proceedings of 2009 IEEE symposium on computational intelligence in scheduling (CISched 2009), Nashville, Tennessee, USA, 30 Mar–2 Apr, pp 90–94
39. Sabar NR, Ayob M, Kendall G, Qu R (2009a) Roulette wheel graph colouring for solving examination timetabling problems. In: Proceedings of the 3rd international conference on combinatorial optimization and applications. Lecture notes in computer science, vol 5573. Springer, Berlin, pp 463–470
40. Thompson J, Dowsland K (1998) A robust simulated annealing based examination timetabling system. Comput Oper Res 25:637–648
41. Yang Y, Petrovic S (2005) A novel similarity measure for heuristic selection in examination timetabling. In: Burke EK, Trick M (eds) Practice and theory of automated timetabling V: selected papers from the 5th international conference. Lecture notes in computer science, vol 3616. Springer, Berlin, pp 377–396