Adaptive Automated Construction of Hybrid Heuristics for Exam Timetabling and Graph Colouring Problems

Rong Qu[1], Edmund K. Burke[1] and Barry McCollum[2]
[1] Automated Scheduling, Optimisation and Planning (ASAP) Group
School of CSiT, University of Nottingham, Nottingham, NG8 1BB, U.K.
rxq@cs.nott.ac.uk, ekb@cs.nott.ac.uk
[2] School of Computer Science, Queens University, Belfast
University Road, N. Ireland, BT7 1NN
b.mccollum@qub.ac.uk

Abstract. In this paper we present a random iterative graph based hyper-heuristic to produce a collection of heuristic sequences to construct solutions of different quality. These heuristic sequences can be seen as dynamic hybridisations of different graph colouring heuristics that construct solutions step by step. Based on these sequences, we statistically analyse the way in which graph colouring heuristics are automatically hybridised. This, to our knowledge, represents a new direction in hyper-heuristic research. It is observed that spending the search effort on hybridising Largest Weighted Degree with Saturation Degree at the early stage of solution construction tends to generate high quality solutions. Based on these observations, an iterative hybrid approach is developed to adaptively hybridise these two graph colouring heuristics at different stages of solution construction. The overall aim here is to automate the heuristic design process, which draws upon an emerging research theme which is concerned with developing computer methods to design and adapt heuristics automatically. Experimental results on benchmark exam timetabling and graph colouring problems demonstrate the effectiveness and generality of this adaptive hybrid approach compared with previous methods on automatically generating and adapting heuristics. Indeed, we also show that the approach is competitive with the state of the art human produced methods.

Keywords: adaptive, exam timetabling, graph colouring, graph colouring heuristics, hybridisation, hyper-heuristic

1 Introduction

Since the 1960s exam timetabling has been one of the most studied subjects in timetabling research (see [20, 40]). This is partly because it is one of the most important administrative activities that take place several times a year in all academic institutions. In the literature, there is a range of survey papers that overview different aspects of educational timetabling research (e.g. [2, 8, 11, 17, 20, 24, 32, 33, 37, 40, 43, 45]).

In a general exam timetabling problem, a number of exams must be scheduled to a limited number of time periods (timeslots). In doing this, some constraints must be satisfied in any circumstances (so called *hard constraints*). In addition, there is also a set of desirable constraints (so called *soft constraints*), which may be violated when no solutions can be found which satisfy all of them. These constraints are usually different from one institution to another. Solutions with no violations of hard constraints are called *feasible* solutions. How much the soft constraints are satisfied gives an indication of how good the solutions (timetables) are.

In a simplified timetabling problem, if we are only concerned with hard constraints, the problem can be represented by a graph colouring model. Vertices in the graph represent exams in the problem, and edges representing the conflicts between exams (i.e. with common students). The problem is to minimise the colours used to colour all vertices, while avoiding the assignment of two adjacent vertices to the same colour. Graph colouring problems are among the most important problems in graph theory and are known to be NP-hard [30].

Graph colouring heuristics such as those presented in [4] were widely studied in early timetabling research [11, 20], and are still being employed today as either the initialisation method for meta-heuristics, or they are being integrated with meta-heuristics in different ways (e.g. [6, 13, 16, 22, 26, 38]). Meta-heuristics [9, 27] have received significant attention in the last two decades and have been very successful over a range of complex timetabling problems [40]. These include Tabu Search (e.g. [25]), Simulated Annealing (e.g. [44] and Evolutionary Algorithms (e.g. [14]).

New techniques and methodologies have also been developed in recent timetabling research. For example, Variable Neighbourhood Search and Very Large Scale Neighbourhood Search have been applied successfully to exam timetabling problems (e.g. [1, 35, 39]) by employing different neighbourhood structures during the search. Iterative techniques such as GRASP have also obtained some success on exam timetabling (e.g. [22, 39]). Other new technologies investigated include Case-Based Reasoning e.g. [12, 18], fuzzy reasoning (e.g. [31]) and hybrid approaches (e.g. [6, 15, 19, 23, 31, 34]).

Hyper-heuristics have received some recent attention in the literature. A hyper-heuristic can be thought of as *a heuristic to choose heuristics* [7]. A set of *low level* heuristics (rather than solutions) represents the search space. One of the motivations is to raise the level of generality of search methodologies, as problem specific information can be restricted to the low level heuristics that deal with the problem solutions directly. This is fundamentally different from most studies of meta-heuristics, where problem specific information is directly incorporated into the design of the algorithms. Approaches that are fine-tuned for particular problems in this way may not work well on different problems, or even different instances of the same problem. Low level heuristics investigated in hyper-heuristic research may be automatically switched in the hyper-heuristic framework to adapt to different problems. In the literature, these include both moving strategies (e.g. [3, 10, 18, 36, 41]) and constructive strategies (e.g. [6, 13, 18, 39]). Graph heuristics are the most widely studied low level constructive heuristics and have provided promising results on a number of timetabling problems.

Adaptive techniques have been studied recently in timetabling research. In an iterative adaptive method developed in [16], orderings of exams by graph heuristics are adapted iteratively to construct solutions by moving forward those exams that are found to be *difficult* to schedule in previous iterations. In this paper, we develop an iterative approach that hybridises graph heuristics adaptively. The ordering of exams to be used to construct solutions can also be seen as being adaptively, but not directly, adjusted. This draws upon earlier work on squeaking wheel optimisation [29]. We adaptively call different ordering strategies during the solution construction process. These heuristics are then hybridised and used to deal with actual solutions directly. Note that the goal of this paper is not to design another heuristic (or meta-heuristic) methodology to compare with the other human designed methods in the literature. Rather, the goal is to present a more effective automated way of designing and adapting heuristics.

2 Benchmark Exam Timetabling and Graph Colouring Problems

The benchmark exam timetabling and graph colouring problems we used to analyse heuristic sequences, and to test the adaptive hybrid approach (see Section 5) were firstly introduced in 1996 [21], and are publicly available at ftp://ftp.mie.utoronto.ca/pub/carter/testprob/ and http://www.cs.nott.ac.uk/~rxq/data.htm. During the last ten years the datasets have been widely tested by a number of approaches in the literature. However, there has been an issue with different instances circulating under the same name. Thus, the datasets were clarified and renamed as version *I*, version *II* and version *IIc* in [40]. We used all versions of the data [40] and present the characteristics of these problems in Table 1. For the problem instance "yor83 IIc", it is observed that no feasible solutions can be obtained by using the GHH approach hybridising LWD, LE or LD. This might be caused by the corrections that were made on "yor83 II", leading to the corrected "yor83 IIc" too hard to solve. Thus in the rest of the paper the results are not presented for this instance. The *conflict density* in Table 1 can be defined as the density of element c_{ij} of value 1 in the matrix, if we define a conflict matrix C, where c_{ij}=1 if exams i and j conflict, c_{ij}=0 otherwise. During the years, variants *a* (exam

timetabling) and *b* (graph colouring) of these benchmarks have widely been tested in the literature. See more details in [40] and http://www.asap.cs.nott.ac.uk/resources/data.shtml.

Table 1 Characteristics of the benchmark timetabling problems in [21], also see [40]

	car91	**car92**	**ear83 I**	**ear83 IIc**	**hec92 I**	**hec92 II**	**kfu93**	**lse91**
No. of exams	682	543	190	189	81	80	461	381
No. of timeslots	35	32	24	24	18	18	20	18
No. of students	16925	18419	1125	1108	2823	2823	5349	2726
Conflict density	0.13	0.14	0.27	0.27	0.42	0.42	0.6	0.6
	sta83 I	**sta83 IIc**	**tre92**	**ute92**	**uta92 I**	**uta92 II**	**yor83 I**	**yor83 IIc**
No. of exams	139	138	261	184	622	638	181	180
No. of timeslots	13	35	23	10	35	35	21	21
No. of students	611	549	4360	2750	21266	21329	941	919
Conflict density	0.14	0.19	0.18	0.8	0.13	0.12	0.29	0.30

In variant *b* of the problem, a set of (80-682) exams need to be scheduled into a limited number of (10-35) timeslots in different instances. Hard constraints avoid students taking two exams at the same time. That is ($i \neq j$ and $D_{ij} > 0) \Rightarrow t_i \neq t_j$; ($D_{ij}$: the number of students in both exams i and j; t_i: the timeslot that exam i is scheduled into). Soft constraints are concerned with spreading the exams taken by students evenly over the timetable. That is, exams with common students should not be assigned to timeslots that are too close to each other (i.e. less than 5 timeslots apart). The penalty of the solution is calculated by using the evaluation function presented in Equation 1.

$$\sum_{k=1}^{m-1}\sum_{l=k+1}^{m} (w_i \times s_{kl}) / S \text{ , } i \in \{0, 1, 2, 3, 4\}$$

where

s_{kl} is the number of students involved in both exams e_k and e_l, if $i = |t_l - t_k| < 5$;

$w_i = 2^{|4-i|}$ is the cost of assigning two conflicted exams e_l and e_k with i timeslots apart, if $i = |t_l - t_k| < 5$, i.e. , i.e. $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, $w_5 = 1$; t_l and t_k as the timeslots of exam e_l and e_k, respectively

m is the number of exams in the problem

S is the number of students in the problem

Equation 1 Evaluation function for the benchmark exam timetabling problems in [21]

The variant *a* represents graph colouring problems. They consist of assigning a minimal number of colours to 81-682 vertices in the graphs while avoiding assigning the same colour to adjacent vertices. If considered in the context of a timetabling scenario, this problem can be seen to minimise the number of timeslots to accommodate all exams into a tight/short timetable. As the number of students enrolled in the exams (vertices) and the number of students evolved in conflicted exams (edges) are still playing a role in the problem, this variant of the benchmark can be seen as a collection of specialised graph colouring problems with weighted vertices and edges. There is no soft constraint in the problem. The number of colours is usually used as the evaluation of the colourings obtained.

3 The Graph Based Hyper-heuristic (GHH)

Graph colouring heuristics on their own are simple constructive techniques where items in the problem are ordered and used one by one to construct solutions. For example, by using Largest Degree (see Table 2), vertices in graph colouring problems are ordered by the number of vertex degrees decreasingly, and are assigned colours one by one. In exam timetabling problems, more information can be employed to decide the

order of the exams to be scheduled (i.e. by the number of students they have, see Largest Enrolment in Table 2), and schedule them one by one to construct timetables. The overall strategy is that the most difficult items in the problems are dealt with first to construct good quality solutions.

Table 2 Ordering strategies used as difficulty measures in graph heuristics in timetabling

Graph heuristics	**Ordering strategies that order the events in the problem**
LD (Largest Degree)	Decreasingly by the number of conflicts the event has with the others (i.e. two events have common students thus are conflicted)
LWD (Largest Weighted Degree)	The same as Largest Degree but weighted by the number of students involved in the conflicted events
LE (Largest Enrolment)	Decreasingly by the number of enrolments in the event
SD (Saturation Degree)	Increasingly by the number of valid timeslots left for the event in the partial timetable
CD (Colour Degree)	Decreasingly by the number of conflicts the event has with those already scheduled

In our previous work on hyper-heuristics [13], heuristic sequences consisting of the five graph heuristics presented in Table 2 and a random ordering strategy were addressed by a standard Tabu Search approach and were used to construct solutions for exam and course timetabling problems. At each step of the Tabu Search, one incumbent heuristic sequence is used to generate one solution. The quality of the solution constructed by this corresponding heuristic sequence is used as the objective value to guide this step of the Tabu Search. Figure 1 presents the pseudo-code of this graph based hyper-heuristic (GHH).

The solution construction at each step of the Tabu Search, using a heuristic sequence (of length e, which is the number of exams in the problem), is an iterative process where, at the i^{th} iteration, the i^{th} heuristic in the sequence is used to order the events (courses or exams) that are not scheduled yet at that iteration. The first event in the ordered list is then scheduled to the timeslot that leads to the least cost in the timetable. This is made by calculating the cost on soft constraint violations, and ties are broken by choosing the first timeslots leading to the least cost. In the $(i+1)^{th}$ iteration, the remaining events are re-ordered by the $(i+1)^{th}$ heuristic, and the first event in the updated list is scheduled to the partial solution built from the previous i iterations.

```
initialisation of the heuristic sequence hl
//Tabu Search upon heuristic sequences
for i = 0 to i = the number of iterations
   h = change two heuristics in heuristic sequence hl
   if h is not in the tabu list
      construct a solution c using heuristic sequence h (see Figure 2)
      if c is complete, and its penalty < the least penalty c_g obtained
           save the best solution, c_g = c
           update the tabu list
           hl = h //a move in Tabu Search
      else backtrack
   //end if
//end of Tabu Search
output the best solution with the penalty c_g
```

Figure 1 The graph based hyper-heuristic with a high level Tabu Search [13]

Figure 2 presents an illustrative example of the solution construction process for a simple problem with six events e1, …, e6 represented by vertices, conflicts between which are represented by edges in the graph. Assume at a certain step of Tabu Search, the heuristic sequence of length 6 is "LD LD LD SD LD SD". A partial solution has been built iteratively by using the first three heuristics in the sequence, leaving the shaded events as not yet scheduled. At the 4th iteration of the solution construction, the 4th heuristic SD in the sequence is used to order the remaining events as "e1, e5, e6" increasingly by the number of valid remaining timeslots in the timetable for them (see SD in Table 2). e1 is then scheduled into the partial solution at the 4th

iteration. At the 5th iteration, LD is used to re-order the remaining events as "e5, e6" (see LD in Table 1), and e5 is scheduled. This process is repeated until all six events are scheduled.

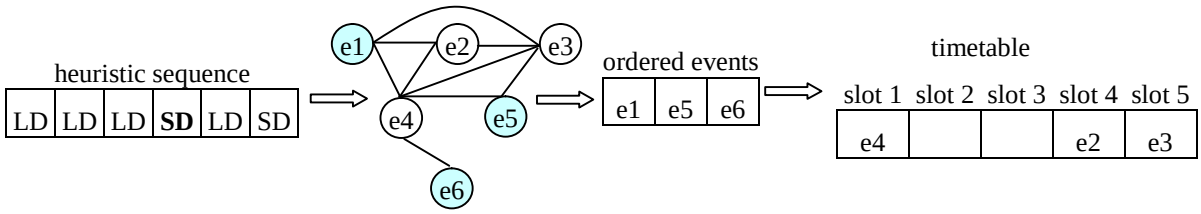


Figure 2 An illustrative example of solution construction using a sequence of graph heuristics

For some heuristic sequences, if at a certain iteration of solution construction the event cannot be scheduled to any feasible timeslot due to its conflict with those already scheduled, this heuristic sequence is then discarded and Tabu Search backtracks and moves to another heuristic sequence to construct another solution.

In [13], the role of Tabu Search was simply to search for the best heuristic sequences without considering the details of actual solutions. The search is thus upon a search space of heuristic sequences, and moves are made by the performance of the low level heuristic sequences on constructing solutions. This is different from most standard meta-heuristics, where concrete solutions and problem dependant constraints are directly considered by the search. A meta-heuristic was first defined by Glover and Laguna as "a master strategy that guides and modifies other heuristics" (page 17 in [28]). The *heuristic* here usually refers to the mechanism of moves "for transforming one *solution* into another" [28], rather than moving between heuristic sequences. Most meta-heuristics in the literature operate directly on a search space of solutions but a hyper-heuristic operates on a search space of heuristics. Here we are using Tabu Search (a meta-heuristic) as a hyper-heuristic.

In GHH, a set of heuristics (rather than solutions themselves) are considered for building solutions. This can be seen as adaptively calling appropriate heuristics during the solution construction. GHH thus can be seen as being able to, at a higher level, adaptively hybridise different heuristics. However, this hybridisation is more on a random basis with the only guidance of solution quality obtained by the corresponding heuristic sequences at the higher level. In this paper, we extend the previous GHH approach [13] with the different focus of developing an adaptive approach where heuristics are dynamically hybridised during solution construction. It is based upon the observations and statistical analysis over a large number of different heuristic sequences obtained by a random iterative GHH. The same solution construction process that was described above will be used in the approaches developed in this paper but the high level approaches are adaptive methods rather than Tabu Search.

4 Hybridisations of Graph Colouring Heuristics within GHH

4.1 A Random Iterative GHH

A random iterative GHH is first developed to iteratively generate heuristic sequences of different quality for the benchmark exam timetabling and graph colouring problems described above in Section 2. Figure 3 presents the pseudo-code of this random iterative GHH. At each step, a sequence of graph colouring heuristics is randomly generated and employed to construct a solution. Those sequences that cannot generate feasible solutions will be discarded as we are only interested in good heuristic sequences. After a certain number of steps (e×50), a large collection of heuristic sequences and the penalties of their corresponding (feasible) solutions are obtained for further analysis on the characteristics of good hybridisations of graph heuristics, based on which an adaptive hybrid approach was developed and is presented in Section 5.

```
for i = 0 to i = e × 50, e: the number of exams/vertices
    for n = 1 to n = e
    initialise heuristic sequence h = {SD SD … SD SD}
    h = randomly change n heuristics in h to LD, LWD or LE
    construct a solution c using h (see Figure 2)
    if solution c is feasible
       save h and the penalty of its corresponding solution c
```

Figure 3 The pseudo-code of the random iterative graph based hyper-heuristic

In this work, to investigate clearly how heuristics are hybridised, we study heuristic sequences consisting of two graph colouring heuristics. It was observed in the literature [16, 21] and in our previous work [13] that when being employed on its own, SD performs the best in most cases due to its ability to *dynamically* order the events according to the number of remaining valid timeslots. However, its efficiency also varies (i.e. other graph colouring heuristics occasionally outperform SD on specific problems). Thus we use SD as the basic heuristic in heuristic sequences (i.e. the initial sequence only contains SD in Figure 3). The other heuristics (LD, LWD and LE in Table 2) are randomly hybridised into the list of SD. CD in Table 2 was found not to be as effective as other heuristics and also very time consuming [13], so it is not considered here.

To guarantee a full coverage of different rates of hybridisation, the random iterative GHH systematically hybridises n LWD, LE or LD, n = [1, …, e], in the sequences. For each amount of LWD, LE or LD hybridisations, 50 samples are obtained. Note that, at the first iteration of solution construction, SD always returns the same level of difficulty (number of valid timeslots/colours) for all exams/vertices. For this reason, at the start, LD, LE or LWD will be employed i.e. the heuristic sequences always begin with LD, LWD or LE rather than SD.

4.2 Analysis of Heuristic Sequences

The random iterative GHH generates a collection of heuristic sequences hybridising different rates of LWD, LE or LD. The idea is to give each rate of hybridisation an even chance and observe the properties of the best heuristic sequences in assisting the adaptive approach to intelligently hybridise different heuristics.

To analyse how these heuristics are hybridised with SD, all these heuristic sequences are stored in Microsoft Excel spreadsheets and ranked by their corresponding solution quality. For example, one row in the Excel spreadsheet could be “LD LD LD SD … SD 11” (see Figure 4), meaning that this sequence generates a solution of penalty 11 using the evaluation function in the problem (either exam timetabling or graph colouring problems). Then, based on the processed data, the analysis of heuristic sequences is carried out in three steps, which are described as follows:

I. Firstly, all the heuristic sequences are grouped into three subsets by their corresponding solution quality: those generating the best 5% solutions, the worst 5% solutions, and those not in these two subsets.

II. Then, the average appearances of LD, LWD or LE at each position of the sequences are calculated for each subset. This gives a number from [0, 1], which is an estimated rate that LD, LWD or LE is hybridised at the particular position in the heuristic sequences in each subset. Assume that, in Figure 4, we have 5 sequences of the best quality for a problem. This will result in an average appearance of LD for each position of "0.8 1 0.2 ... 0.4" in the heuristic sequences. This shows that LD when employed more often at the second and third positions (i.e. the early stage of solution construction) tends to generate the best solutions.

III. Finally, trendlines of LD, LE or LWD hybridisations in heuristic sequences are plotted by using Microsoft Excel regression analysis tools. Due to the amount of fluctuation in the hybridisations of LD, LE or LWD, polynomial trendlines of order 4-5 have been used to plot a “best fit” curve based on the

appearances of the LD, LE or LWD at different positions of solution construction (of the three subsets of different quality) respectively.

Sequence 1	LD	LD	LD	SD	…	…	…	…	SD	11
Sequence 2	LD	LD	LD	SD	…	…	…	…	SD	11.2
Sequence 3	LD	LD	LD	LD	…	…	…	…	LD	11.5
Sequence 4	LD	LD	LD	SD	…	…	…	…	SD	12.5
Sequence 5	LD	SD	LD	SD	…	…	…	…	SD	12.8
Hybridisation Rate		0.8	1	0.2	…	…	…	…	0.2	

Figure 4 Rate of hybridisations of LD with SD at different positions in heuristic sequences

Note that the "hybridisation rate" in Figure 4 is different from the overall amount of LWD, LE or LD hybridisations (*n* in heuristic sequences, see Figure 3). The former represents the rate at which LWD, LE or LD appears at particular positions of heuristic sequences. The latter represents the overall number of LWD, LE or LD appearance in a whole heuristic sequence.

4.2.1 Heuristic Sequences for Exam Timetabling Problems

We first collect the heuristic sequences hybridising LD, LE or LWD with SD by carrying out the random iterative GHH on the benchmark exam timetabling problems described in Section 2. Table 3 presents the penalties of the best and worst solutions obtained by these sequences. The corresponding overall amounts of LD, LE or LWD hybridised (% of LD, LE or LWD in the sequences) are also presented.

Table 3 Penalties of the best and worst solutions from the heuristic sequences hybridising different amount of LD, LE or LWD, respectively, by the random iterative GHH (RGH) for benchmark exam timetabling problems. The best results are in bold. "% of Lx" gives the average overall amount of LWD, LD or LE hybridised in heuristic sequences.

	RGH-LE best		**RGH-LE worst**		**RGH-LD best**		**RGH-LD worst**		**RGH-LWD best**		**RGH-LWD worst**	
	best	%	worst	%	best	%	worst	%	best	%	worst	%
car91	5.56	29	6.85	27	5.43	29	6.39	23	**5.26**	36	6.06	25
car92	4.75	11	5.96	32	4.47	34	5.77	30	**4.43**	29	5.2	18
ear83 I	41.06	14	52.66	32	40.24	38	49.86	14	**37.95**	47	49.07	19
ear83 IIc	45.70	3	54.55	14	45.10	32	50.61	14	**40.74**	35	50.01	12
hec92 I	13.33	34	21.63	37	12.44	26	16.74	35	**12.15**	53	15.28	9
hec92 II	13.23	10	17.25	23	12.35	22	17.87	44	**12.26**	59	15.26	15
kfu93	16.05	24	21.76	37	16.06	40	21.01	43	**15.37**	36	20.27	10
lse91	12.2	10	17.14	44	12.41	42	16.59	45	**12.01**	23	15.23	12
sta83 I	165.53	26	184.93	36	163.18	10	185.02	39	**159.58**	34	180.47	31
sta83 IIc	34.52	14	46.14	37	34.7	16	40.13	67	**34.31**	27	40.74	35
ute92	29.38	12	38.56	45	29.88	11	38.6	24	**28.98**	63	34.38	10
tre92	9.48	15	11.83	26	8.87	30	11.63	30	**8.76**	34	11.09	19
uta93 I	4.18	18	5.29	28	4.13	26	5.37	15	**3.95**	29	4.95	17
uta93 II	3.8	3	4.6	7	3.69	6	4.42	7	**3.62**	7	3.92	3
yor83 I	44.45	13	51.67	27	**41.72**	10	48.81	23	42	27	48.15	19

It is clear, from Table 3, that sequences hybridised with LWD performed the best for almost all the exam timetabling problems. One possible reason may be that LWD can be seen as integrating both LE (the number of students) and LD (the number of conflicts) in an intelligent way.

We present, in Figure 5, the trends of LWD appearances in the best 5% heuristic sequences for three benchmark problem instances (hec92 I, ute92 I and car92 I). The trends of hybridisation for the rest of the problem instances are presented in Appendix A. In generating the trends, the first heuristics are always ignored as they are fixed as LE, LD or LWD. The overall observation from these trends is that in most of the problems tested, the best heuristic sequences employ more LWD at the beginning rather than at the later stages of the solution construction. Another observation is that the hybridisations of LWD vary significantly

at the early positions of sequences (i.e. there are greater changes of LWD and SD at the early stages of solution construction).

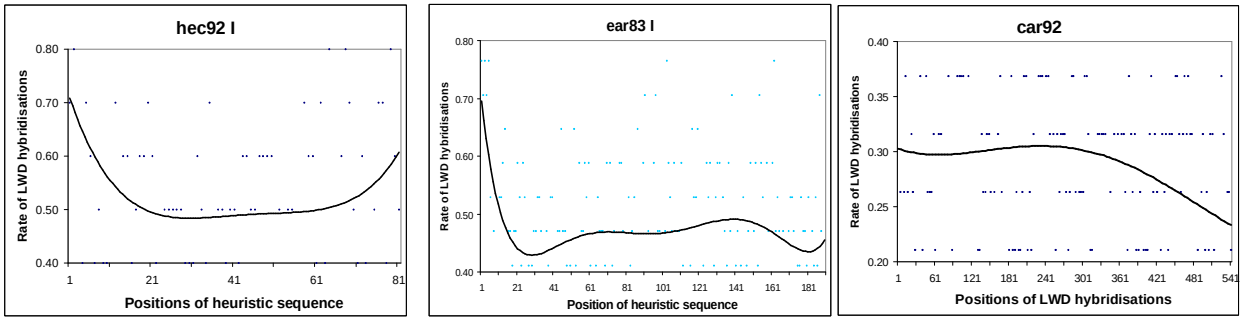


Figure 5 The trends of hybridising LWD in the best heuristic sequences for hec92 I, ear83 I and car92 I

Another observation from Table 3 is that the overall amount of LWD hybridisations in the best heuristic sequences for different instances is quite different. To have a closer look at the hybridisations of LWD, we plot the box-whisker distributions of the hybridisation amount in the whole sequences in Figure 6. For some instances such as sta83 I and ute92, the range of LWD hybridisations is wider than for other instances such as yor83 I and sta83 IIc. This indicates that hybridising a certain amount of LWD needs to be carefully made within the GHH approach for the latter problems.

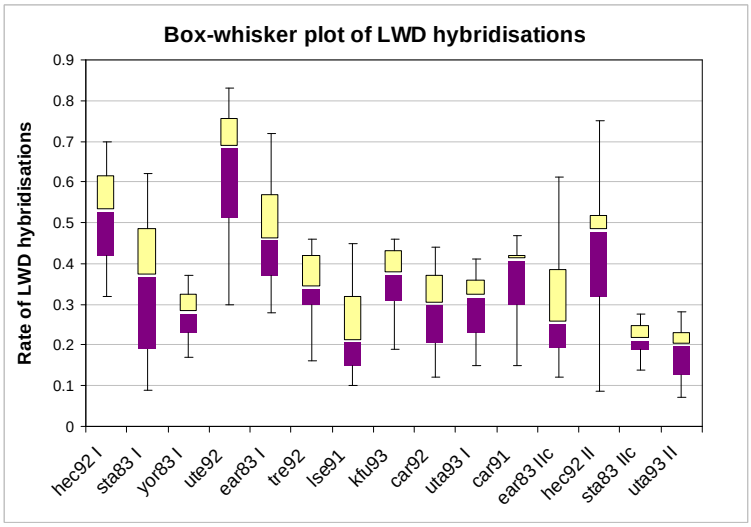


Figure 6 The amount of LWD hybridisation in the best heuristic sequences for exam timetabling problems

For the LE and LD hybridisations within the best heuristic sequences, the observations are that no obvious trends can be obtained in heuristic sequences that generate the best solutions for all the problems (see Figure 7). The worst 5% of heuristic sequences also have different trends on the hybridisations of LD, LE and LWD for different instances and thus no obvious observations are obtained. Due to space limitations, and also because we are not particularly interested in the characteristics of the worst sequences, we do not present these trends in the paper.

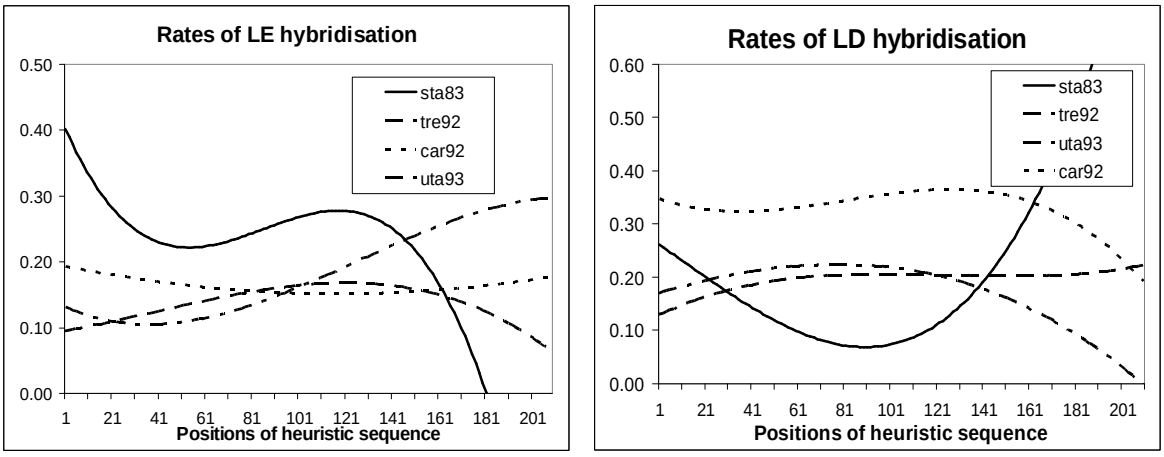


Figure 7 Hybridisation trends of LD and LE in the best heuristic sequences for exam timetabling problems

4.2.2 Heuristic Sequences for Graph Colouring Problems

The same random iterative GHH presented in Figure 3 was run to obtain heuristic sequences for the graph colouring problems presented in Section 2. Due to the generality of the approach, the only differences when applying the random GHH for both timetabling and graph colouring problems is the evaluation function used (i.e. the number of colours used in the colourings rather than the violations of soft constraints in timetables).

The evaluation function we used in the graph colouring problem is simply the number of colours in the colourings. Due to the fact that some different colourings may use the same number of colours, some other evaluation functions have been used in the literature. For example, sum colouring [42] evaluates not only the number of colours used, but also the sum of the chromatic numbers of the colours, aiming to give a more informative evaluation. In our experiments, we found that these two evaluation functions performed in a similar way. This may be because our GHH approaches are constructive, and thus are less dependant on the evaluation functions which can play an important role in the usual implementations of local search algorithms.

As the generated colourings are always feasible, all the heuristic sequences and their corresponding quality value (i.e. the number of colours) are saved for further analysis. Table 4 presents the best and worst results obtained by these sequences for the graph colouring instances. Again, we present trends of the LWD appearance in the best heuristic sequences for three problems in Figure 8, and the rest of the instances in Appendix B.

Table 4 Best and worst results from the heuristic sequences hybridised with LD, LE or LWD, respectively using the random iterative GHH (RGH) for graph colouring problems. The best results are in bold. "% of Lx" gives the average overall amount of LWD, LD or LE hybridised in heuristic sequences.

	car91	car92	ear83	hec92	kfu93	lse91	sta83	tre92	ute92	uta92	yor83
RGH-LD best	**30**	**29**	**22**	**18**	**19**	**17**	**13**	21	**10**	**31**	**20**
% of LD	29	24	15	24	31	19	49	36	10	27	16
RGH-LD worst	38	35	28	21	23	21	13	26	13	35	25
% of LD	93	77	86	38	79	78	49	61	14	79	54
RGH-LE best	**30**	**29**	**22**	**18**	**19**	**17**	**13**	**20**	**10**	**31**	**20**
% of LE	17	18	17	25	36	11	49	16	29	11	10
RGH-LE worst	45	38	32	25	28	27	**13**	30	16	35	29
% of LE	93	82	93	82	98	97	49	95	97	45	94
RGH-LWD best	**30**	**29**	**22**	**18**	**19**	**17**	**13**	**20**	**10**	**31**	**20**
% of LWD	36	13	19	37	46	28	49	45	34	24	17
RGH-LWD worst	37	36	29	22	23	20	13	25	12	35	25
% of LWD	89	92	63	69	16	94	49	15	20	74	78

Again, sequences hybridised with LWD performed the best (although quite slightly) for all instances. For almost all instances tested, the best heuristic sequences employ more LWD at the beginning of solution construction. For some problems in Appendix B (i.e. lse91) there are no obvious trends observed. However, the LWD hybridisations vary more at the early stages. Note that, for problem instance sta83, all the heuristic sequences are of the same quality. The levels of hybridisation of LE and LD within the best heuristic sequences again have no obvious trends and are not presented in the paper.

The range of the overall level of LWD hybridisation in Figure 9 again shows that, for different problems, the rates of hybridisation in the best heuristic sequences are very different. The range of hybridisation levels for some problems (i.e. "ear83" and "car92") is much smaller and indicates that the problems are more difficult to solve using the GHH approach. It can also be seen that these distributions are quite different from those in Figure 6 for exam timetabling problems.

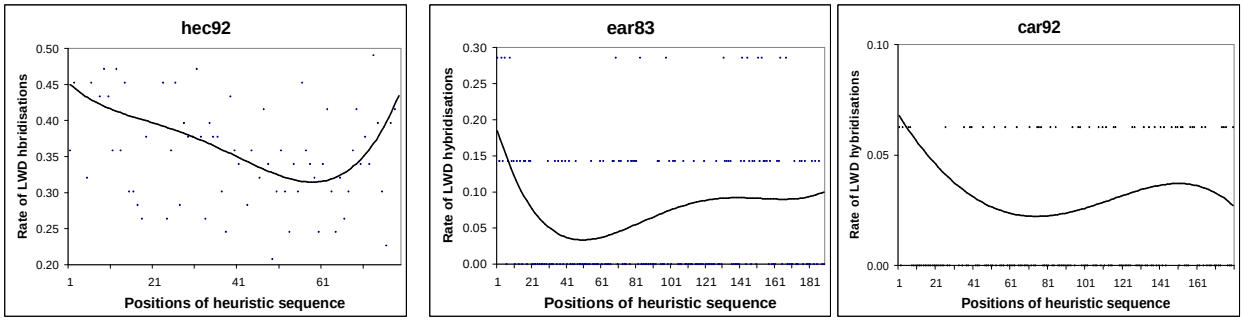


Figure 8 Trends of hybridising LWD in the best heuristic sequences for graph colouring problems car92, hec92 and ear83

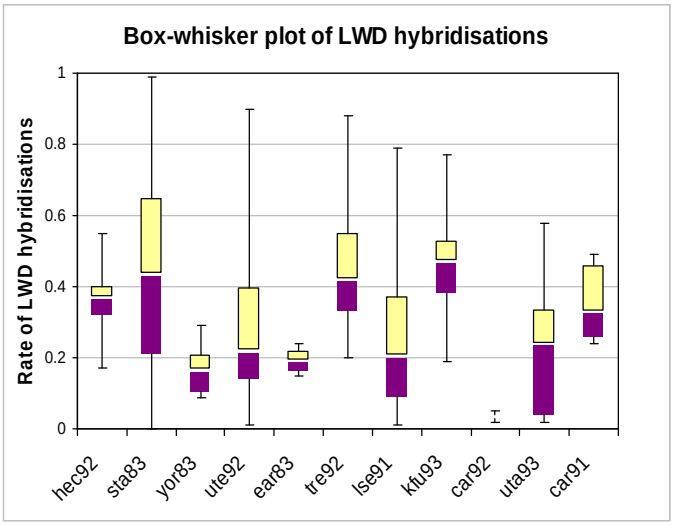


Figure 9 The amount of LWD hybridisations in the best heuristic sequences for graph colouring problems

To summarise, based on the statistic analysis upon a large amount of heuristic hybridisations, it is observed that LWD hybridised with SD generates better quality solutions than hybridising LE or LD for both exam timetabling and graph colouring problems. Furthermore, hybridising more LWD at the early stage of SD heuristic sequences tends to generate the best solutions. However, the rate of LWD hybridisation at the early stage of solution construction varies significantly. For different problems, the overall amount of LWD hybridisation within the whole heuristic sequence is also quite different.

5 Adaptive Hybridisation of Graph Heuristics

The above observations on both exam timetabling and graph colouring problems indicate that although the hybridisations present some trends in the heuristic sequences, the levels of hybridisation and their appropriate ranges vary a lot for different instances. The heuristic hybridisations at the beginning of heuristic sequences also vary, although in general the LWD hybridisations at the early stages are higher. Thus an intelligent approach needs to be developed to adaptively hybridise different *amounts* of LWD at different *stages* of solution construction. The adaptive approach is tested and shown to be effective and comparable with the current best approaches in the literature upon both the benchmark exam timetabling problems and graph colouring problems.

There are two stages in the adaptive approach to hybridise LWD at different parts of the heuristic sequences. The process is presented as follows:

I. In the first stage, LWD is iteratively hybridised into the first half of the heuristic sequences based on SD (i.e. no LWD appears in the second half of heuristic sequences). This is an adaptive process that adjusts the amount of LWD hybridisation in a basic sequence of SD iteratively (see Figure 10), and stops after a certain number of iterations. Our above analysis suggests that the effort of hybridising LWD should be made at the early stage of solution construction. Another reason is that the ordering of events/vertices at later stages of solution construction tends to be less important for constructing good solutions.

II. Based on the best heuristic sequences obtained from stage I, an iterative adjustment is made to hybridise LWD over the whole heuristic sequence. This is because LWD may also contribute to the later stage of solution construction. The process stops after a certain number of iterations.

i) If a better solution is generated from the current sequence, the amount of LWD to be hybridised in the first half of the sequence is increased by $\alpha = 0.03$. The aim is to further explore different hybridisations and avoid converging quickly to a fixed amount. The best heuristic sequences and the best amount are saved and updated in stage II of the adaptive approach. The hybridisation amount is limited within the range [10%, 70%], and will be reset as the current best amount if it reaches one end. Note that a hybridisation amount of 1 will result in a sequence with only LWD in the first half (i.e. LWD...LWD SD...SD).

ii) If the solution obtained is not feasible, the hybridisation amount is increased by 0.03.

iii) If a feasible but worse solution is generated, the hybridisation amount is decreased by $\beta = 0.01$.

Figure 10 Adaptive hybridisation of LWD and SD in stage I of the adaptive approach

Figure 10 presents the pseudo-code of stage I of the approach, where LWD is hybridised in the first half of the heuristic sequences. The parameters have been selected based on both the analysis upon the large amount of data in Section 4.2.1, and our empirical experiments. It was observed through experiments that adjusting the incremental value α within the range of [0.01, 0.03] to hybridise 10%-70% LWD did not affect much the solution quality. However, the computational time is different. For example, a smaller increment causes a larger computational time.

5.1 Adaptive Heuristic Hybridisation for Benchmark Exam Timetabling Problems

We evaluate our adaptive GHH approach by comparing it with two approaches on the benchmark exam timetabling instances. The average computational time across the 16 instances is presented for 30 runs on a Pentium IV machine with 1GB memory. The number of iterations of these three approaches is:

- Random iterative GHH (RGH in Section 4.1): $e\times5$ iterations for large problems (uta92 I, uta92 II, car91 and car92) and $e\times10$ iterations for the other problems, respectively.
- Random GHH with a fixed hybridisation of 20% LWD to SD: $e\times10$ iterations for large problems and $e\times10$ iterations for the other problems, respectively.

- Adaptive GHH: *e* in stage I and *e*×2 in stage II for large problems and *e*×2 in stage I and *e*×5 in stage II for the other problems, respectively.

The results are presented in Table 5. Note that these results are obtained by running the first two approaches for a longer computational time compared with the adaptive GHH approach (*e*×10 or *e*×5 iterations vs. *e*×2+*e*×5 or *e*+*e*×2 iterations).

Table 5 shows that the adaptive approach performs the best on all instances. Comparisons with "RGH 20%" indicate that hybridising LWD adaptively (AGH) rather than at a fixed amount contributes to a better performance. Note that "RGH 20%" requires more computational time. For different problems, the levels of LWD hybridised in the best sequences are within the range of [14%, 66%].

Compared with the random iterative GHH (RGH), the adaptive approach (AGH) is more effective on both solution quality and computational time. This indicates that by concentrating on adaptive heuristic hybridisations at the early stage, and a quick adjustment of the overall sequences afterwards, the adaptive approach can quickly identify the appropriate heuristic hybridisations and thus obtain better results.

Table 5 Results from the adaptive approach (AGH), random GHH with fixed hybridisation (RGH 20%), and random GHH (RGH). There were 30 runs for each approach. Computational time is presented in seconds. The best average and best results are given in italics and in bolds, respectively.

	AGH				**RGH 20%**			**RGH**			
	average	**best**	**%**	**Time (s)**	**average**	**best**	**Time (s)**	**average**	**best**	**%**	**Time (s)**
car91	*5.29*	**5.11**	27	10816	5.55	5.38	23873	5.52	5.37	28	25678
car92	*4.48*	**4.32**	33	13125	4.62	4.5	20478	4.71	4.5	27	20631
ear83 I	*36.68*	**35.56**	50	1304	39.68	39.02	2887	37.54	36.51	45	2677
ear83 IIc	*40.98*	**39.38**	50	802	42.17	41.26	1182	42.37	40.13	45	1048
hec92 I	*11.94*	**11.62**	40	78	12.66	12.19	259	12.44	12.08	52	239
hec92 II	*11.88*	**11.5**	66	97	12.58	11.26	398	12.36	11.95	58	346
kfu93	*15.56*	**15.18**	30	632	15.84	15.6	1037	15.7	15.4	36	952
lse91	*11.47*	**11.32**	16	1009	11.58	11.5	2007	11.58	11.48	23	1710
sta83 I	*15.57*	**158.88**	18	56	159.78	159.08	97	160.02	159.58	35	108
sta83 IIc	*34.45*	**34.32**	14	78	34.57	34.47	106	34.64	34.37	21	123
ute92	*29.02*	**28**	25	69	29.33	29.16	105	29.06	28.69	56	129
tre92	*8.74*	**8.52**	26	633	8.83	8.73	1058	8.87	8.77	34	1012
uta93 I	*3.36*	**3.21**	29	13580	3.89	3.63	18830	3.69	3.4	29	23600
uta93 II	*3.49*	**3.45**	32	9869	3.56	3.53	16026	3.63	3.59	39	17356
yor83 I	*41.73*	**40.71**	58	219	42.18	41.52	463	42.66	41.79	29	801

The distributions of the results from the different approaches are presented in Figure 11 to give more statistical information. For different instances, the performance of the three approaches is different with respect to the distributions of the results. In general, the results of the random GHH fall into a normal distribution, but are much worse than the other two approaches. Random GHH with 20% LWD hybridisations have less deviation and a comparison with the random GHH is not conclusive. Compared with the random GHH and random GHH with 20% LWD hybridisations, the adaptive GHH approach are better upon both average and best results for all instances.

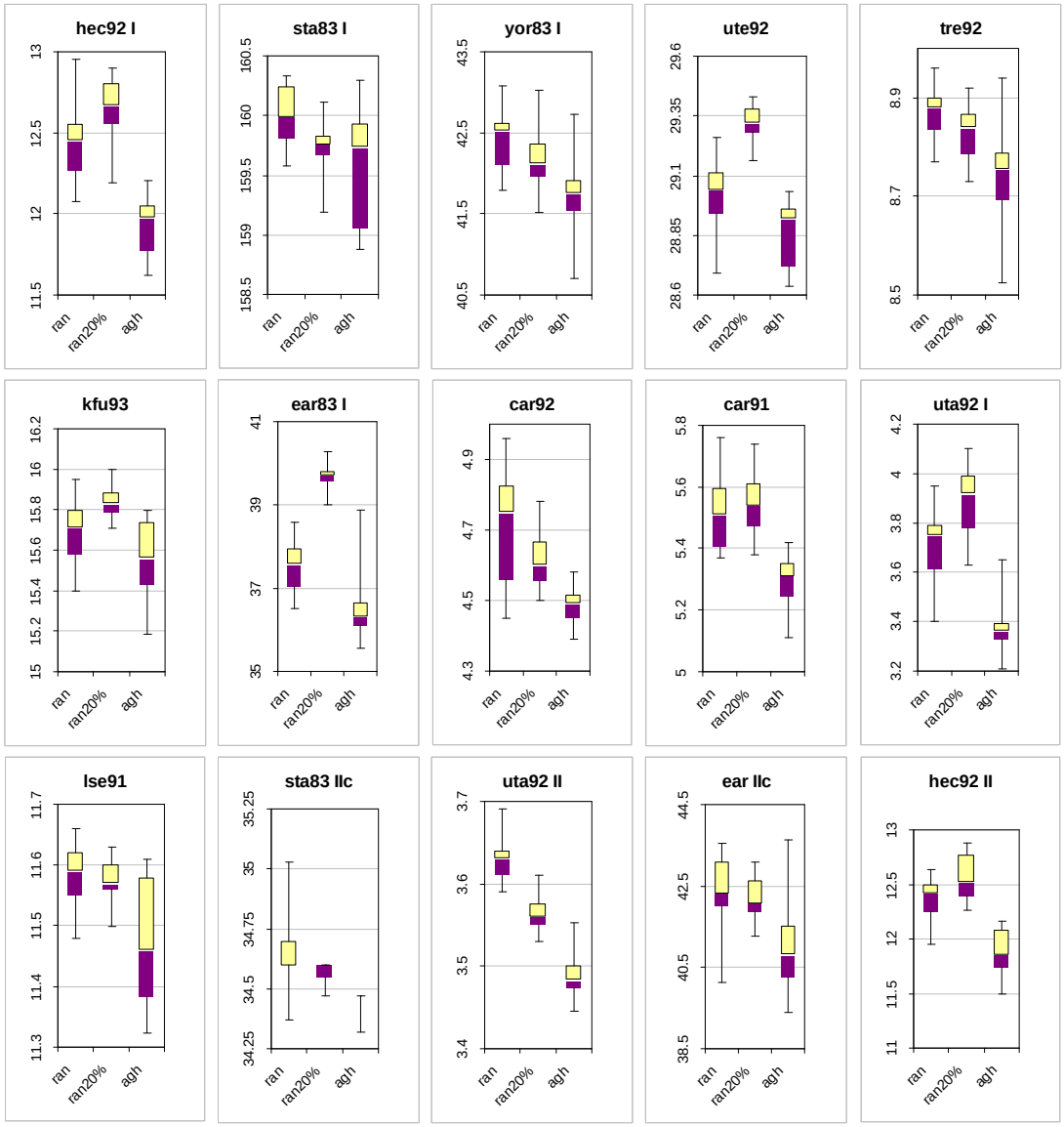


Figure 11 Results of adaptive GHH (agh), random GHH (ran) and random GHH with 20% LWD hybridisations (ran20%) for benchmark exam timetabling instances

A t-test is also carried out to give an indication if the results from the adaptive GHH and random GHH are significantly different. Table 6 summarises the p-values of the t-test. It can be seen that for all problem instances, the results from the random GHH and adaptive GHH approaches are statistically different.

Table 6 t-test on the results from the adaptive GHH and random GHH approaches

	car91	**car92**	**ear83 I**	**ear83 IIc**	**hec92 I**	**hec92 II**	**kfu93**	**lse91**
p-value	1.74E-05	9.2E-05	0.01	0.004	2.8E-06	4.43E-06	0.02	2.1E-03
t Stat	6.3	5.4	2.95	3.43	7.53	7.22	2.66	3.77
	sta83	**sta83 IIc**	**tre92**	**ute92**	**uta92 I**	**uta92 II**	**yor83 I**	
p-value	8.8E-03	0.001	4.1E-04	0.01	2.82E-05	3.5E-10	9.8E-04	
t Stat	3.04	4.0	4.71	2.94	6.08	15.43	4.15	

We also compare our AGH with those in the literature in Table 7, where the best results reported among all the approaches are highlighted. Recall that the aim of the paper is to better understand how we can

automatically hybridise and adapt heuristics. We do not expect to outperform human designed heuristics and meta-heuristics which are tailored specially for this exam timetabling benchmark. However, we can demonstrate that these automatically generated heuristics achieve results that are competitive with the human designed methods. Unfortunately it is not possible to compare the computational time of the different approaches across different platforms because the computational time for most of the approaches in the literature was not reported.

We can also observe from Table 7 that the best results on the 11 problems tested are obtained from different approaches which have been presented in the literature over the years. None of the approaches can be seen as comprehensively outperforming the others. Note that most of the other approaches are specially designed for the particular problems and require initial solutions.

Table 7 The best results from the adaptive approach (AGH), its improvement by the steepest descent method (AGH-SDM), GHH using Tabu Search (TS-GHH) and other approaches in the literature on the benchmark exam timetabling problems.

	car91	**car92**	**ear83 I**	**hec92 I**	**kfu93**	**lse91**	**sta83 I**	**tre92**	**ute92**	**uta92 I**	**yor83 I**
AGH	*5.11*	*4.32*	*35.56*	*11.62*	*15.18*	*11.32*	158.88	*8.52*	*28*	*3.21*	*40.71*
AGH-SDM	*5.09*	*4.26*	*35.48*	*11.46*	*14.68*	*11.2*	158.28	*8.51*	*27.9*	*3.15*	*40.49*
TS-GHH [13]	5.36	4.93	37.92	12.25	15.3	11.33	*158.19*	8.92	28.01	3.88	41.37
[1] (2007)	5.21	4.36	34.87	10.28	**13.46**	10.24	159.2	**8.13**	**24.21**	3.63	**36.11**
[5] (2004)	4.8	4.2	35.4	10.8	13.7	10.4	159.1	8.3	25.7	3.4	36.7
[15] (2003)	**4.65**	**4.1**	37.05	11.54	13.9	10.82	168.73	8.35	25.83	**3.2**	37.28
[16] (2004)	4.97	4.32	36.16	11.61	15.02	10.96	161.91	8.38	27.41	3.36	40.77
[19] (2008)	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	158.2	9.4	24.4	3.5	36.2
[21] (1996)	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	25.8	3.5	41.7
[25] (2000)	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	27.8	4.2	42.0
[34] (2003)	5.1	4.3	35.1	10.6	13.5	10.5	157.3	8.4	25.1	3.5	37.4
[38] (2007)	5.45	4.5	36.15	11.38	14.74	10.85	**157.2**	8.79	26.68	3.55	42.2

The adaptive hybrid approach is an efficient and much simpler method compared with the previous GHH using Tabu Search [13] (see Table 7), which required much more computational time. Not only a larger number of iterations, but also a steepest descent was needed at each step of the Tabu Search based GHH in [13] to further improve each compete solution obtained. This added much more computational expense. The adaptive GHH approach obtained better results on its own (in italics in Table 7) with a much shorter computational time compared with the previous TS-GHH approach. A quick steepest descent on the final solution obtained can further improve the solution quality in seconds (also in italics in Table 7).

For version *II* and version *IIc* of the dataset, there are so far no confirmed results published in the literature. We present the results of our AGH approach in Table 8, with the improved solution penalties after the quick steepest descent.

Table 8 Best results from the adaptive approach (AGH) and its improvement by steepest descent method (AGH-SDM) on version II and version IIc of the benchmark exam timetable instances.

	ear83 IIc	hec92 II	sta83 IIc	uta92 II
AGH	39.38	11.5	34.32	3.45
AGH-SDM	39.36	11.45	33.63	3.44

5.2 Adaptive Heuristic Hybridisation for Graph Colouring Problems

We also test exactly the same adaptive approach on the benchmark graph colouring instances to demonstrate the generality of this method. The results are presented in Table 9. We found that the approach quickly obtained the same or better results in stage I compared with the random iterative GHH in a much shorter time even without further adjustment in stage II. Thus, the stopping condition of the random iterative GHH and

the adaptive approach is set as when the best results are obtained and presented in Table 9 to further demonstrate the efficiency of the adaptive approach.

Table 9 Best results from the adaptive approach (AGH) and random iterative GHH (RGH) on the graph colouring instances. The best results reported in the literature are also presented. 30 runs were employed for the AGH approach.

	car91	car92	ear83	hec92	kfu93	lse91	sta83	tre92	ute92	uta92	yor83
AGH best	30	29	**22**	**17**	**19**	**17**	**13**	**20**	**10**	31	**19**
% LWD	15	16	15	16	30	30	25	15	30	30	17
Time (s)	2814	1578	29	3	5	2	0.4	42	0.4	57	117
RGH best	30	29	**22**	18	**19**	**17**	**13**	**20**	**10**	31	20
% LWD	36	13	19	37	46	28	49	45	34	24	17
Time (s)	4798	2378	55	187	167	2	0.4	79	0.5	1307	3333
Best reported [40]	**28**	**28**	**22**	**17**	**19**	**17**	**13**	**20**	**10**	**30**	**19**

We can see that for 3 instances, the two approaches take the same time to obtain the best results. For the rest of the instances, the random iterative GHH needs much more time to obtain the same or a worse result compared with the adaptive approach. The amount of the LWD hybridised in AGH is also presented in Table 9, indicating that the adaptive approach can quickly locate the correct range of hybridisation, leading to the same or better results compared with the random iterative GHH.

Compared with the best results reported from different approaches developed in the literature, the adaptive GHH obtained competitive results. Note that our adaptive approach is purely constructive, with no further improvement on either solutions or heuristic sequences.

The RGH approach obtained worse results on 3 instances and the same results on the other 8 instances. The difference is that the computational time of RGH is larger to get the same results. Another observation is that within almost all 30 runs, both approaches obtained the same best results for all problem instances. A statistical analysis is thus not presented here.

6 Conclusions

This paper presents an automated heuristic construction approach where Largest Weighted Degree is adaptively hybridised with Saturation Degree at different stages of the solution construction for both exam timetabling and graph colouring problems. This adaptive approach is simple yet effective, and produced comparable results with the best approaches developed during the years in the literature for benchmark instances of both problems. Note that most of the approaches were specifically developed by humans rather than automatically generated (as is the case here).

We presented a statistical analysis of a large collection of heuristic sequences of differing quality. They are obtained by using a random iterative graph based hyper-heuristic methodology and can be seen as hybridising different graph colouring heuristics to construct good solutions. It is not only the case that the general hyper-heuristic can search for appropriate heuristic sequences to generate high quality solution instances for different problems, but also the obtained heuristic sequences can be further analysed for in-depth insight of heuristic hybridisations. Our analysis indicates that the effort should be spent on hybridising Largest Weighted Degree with Saturation Degree at the early stage of solution construction. The development of this adaptive approach draws upon the fact that the hybridisation levels are different for different instances (and that it also varies significantly at the early stage of solution construction).

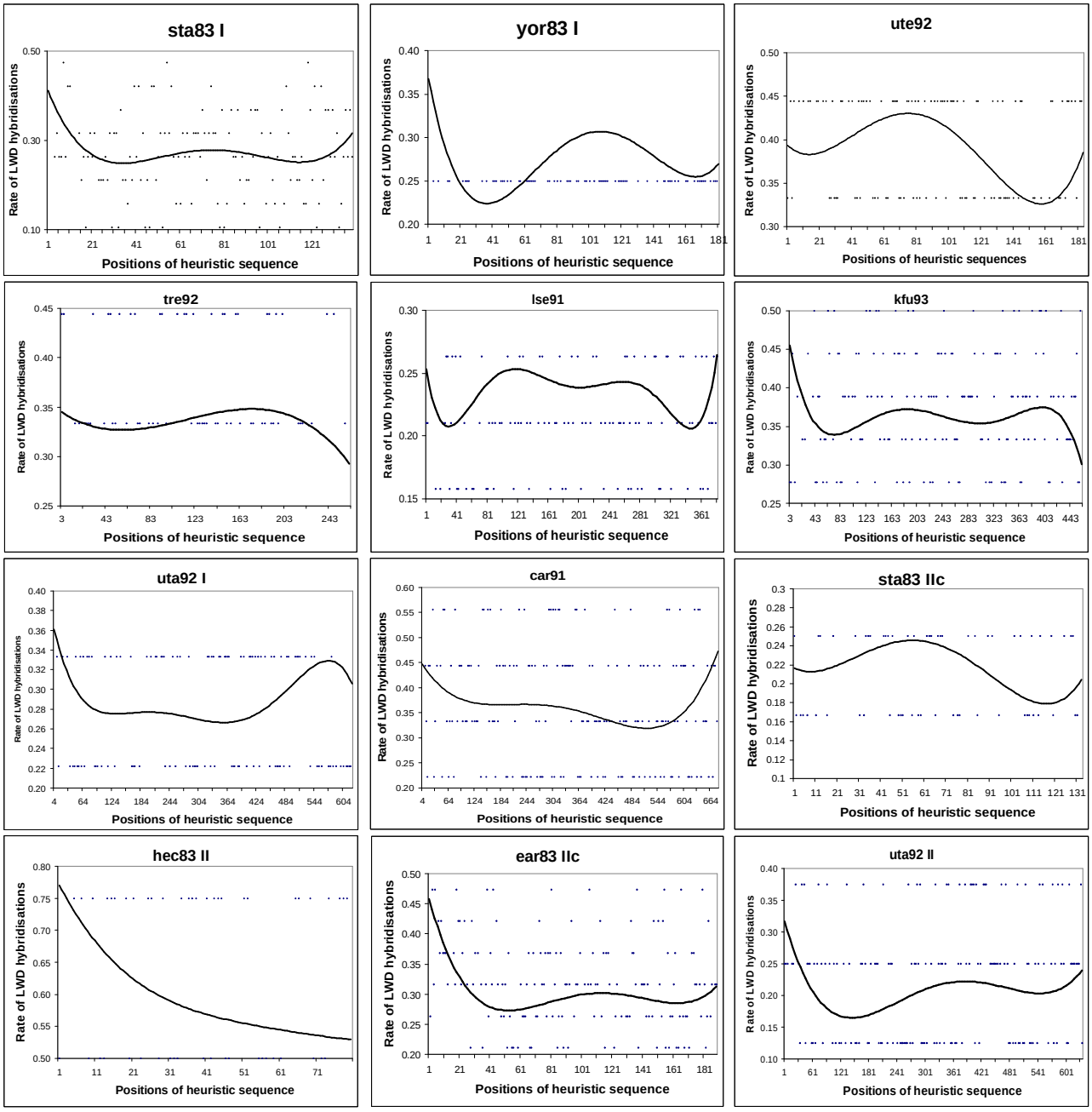
In terms of future research directions, it is interesting to note that as the solutions generated by different heuristic sequences correspond to very different solutions in the solution space [13], it will be beneficial to investigate this approach on generating diversified initial solutions for different meta-heuristic approaches. Future work will also further study different statistical tools to analyse heuristic sequences within hyper-heuristic approaches. The objective is to observe a general mechanism or rules for hybridising different heuristics with the aim of developing more general automated search methodologies.

References

1. Abdullah S., Ahmadi S., Burke E. and Dror M.: Investigating Ahuja-Orlin's Large Neighbourhood Search for Examination Timetabling. *OR Spectrum*, **29**(2): 351-372, 2007.
2. Bardadym V.A.: Computer-aided School and University Timetabling: The New Wave. In: Burke E.K. and Ross P. (eds.): Practice and Theory of Automated Timetabling I: Selected Papers from the 1st International Conference. Lecture Notes in Computer Science, vol. 1153. 22-45, 1996.
3. Bilgin B., Ozcan E. and Korkmaz E.E.: An Experimental Study on Hyper-Heuristics and Exam Scheduling. Proc. of the 6th International Conference on the Practice and Theory of Automated Timetabling, 123-140, 2006.
4. Brelaz D. New Methods to Color the Vertices of a Graph. *Communications of the ACM*, **22**(4): 251-256, 1979.
5. Burke, E., Bykov, Y., Newall, J. and Petrovic S.: A Time-Predefined Local Search Approach to Exam Timetabling Problems. *IIE Transactions*. **36**(6): 509-528, 2004.
6. Burke E.K., Dror M., Petrovic S. and Qu R.: Hybrid Graph Heuristics within a Hyper-heuristic Approach to Exam Timetabling Problems. In: Golden B.L., Raghavan S. and Wasil E.A. (eds.). *The Next Wave in Computing, Optimization, and Decision Technologies*. 79-91. Springer, 2005.
7. Burke E.K., Hart E., Kendall G., Newall J., Ross P. and Schulenburg S.: Hyperheuristics: an Emerging Direction in Modern Search Technology. In: Glover F. and Kochenberger G.: *Handbook of Metaheuristics*, 457-474, 2003.
8. Burke E.K., Jackson S., Kingston H. and Weare F.: Automated Timetabling: the State of the Art. *The Computer Journal*. **40**(9): 565-571, 1997.
9. Burke E.K. and Kendall G. (eds.): *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*. Springer. 239-272, 2005.
10. Burke E.K., Kendall G. and Soubeiga E.: A Tabu Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*. **9**(6): 451-470, 2003.
11. Burke E.K., Kingston J. and de Werra D.: Applications to Timetabling. In: Gross J. and Yellen J. (eds.), Handbook of Graph Theory. Chapman Hall/CRC Press. 445-474, 2004.
12. Burke E.K., MacCarthy B., Petrovic S. and Qu R.: Multiple-Retrieval Case-Based Reasoning for Course Timetabling Problems. *Journal of Operational Research Society*, **57**(2): 148-162, 2006.
13. Burke E.K., McColum B., Meisels A., Petrovic S. and Qu, R.: A Graph-Based Hyper Heuristic for Timetabling Problems. *European Journal of Operational Research*, **176**: 177-192, 2007.
14. Burke E.K. and Newall J.P.: A Multi-stage Evolutionary Algorithm for the Timetable Problem. *IEEE Transactions on Evolutionary Computation*, **3**(1): 63-74, 1999.
15. Burke E.K. and Newall J.: Enhancing Timetable Solutions with Local Search Methods. In: Burke E.K. and Causmaecker P. (eds.): Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference, Lecture Notes in Computer Science 2740, 195-206, 2003.
16. Burke E.K. and Newall J.P.: Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings. *Annals of operations Research*, **129**: 107-134, 2004.
17. Burke E.K. and Petrovic S.: Recent Research Directions in Automated Timetabling. *European Journal of Operational Research*, **140**(2): 266-280, 2002.
18. Burke E.K., Petrovic S. and Qu R.: Case Based Heuristic Selection for Examination Timetabling. *Journal of Scheduling*, **9**(2): 115-132, 2006.
19. Caramia M., Dell'Olmo P. and Italiano G.F.: Novel Local-Search-Based Approaches to University Examination Timetabling. *INFORMS Journal of Computing*, **20**(1): 86-99, 2008.
20. Carter M. and Laporte G.: Recent Developments in Practical Exam Timetabling. In: Burke E.K. and Ross P. (eds.): Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference (PATAT95), Lecture Notes in Computer Science 1153, 3-21, 1996.
21. Carter M., Laporte G. and Lee S.: Examination Timetabling: Algorithmic Strategies and Applications. *Journal of Operational Research Society*, **47**: 373-383, 1996.
22. Casey S. and Thompson J.: GRASPing the Examination Scheduling Problem. In: Burke E.K. and Causmaecker P. (eds.): Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference (PATAT02), Lecture Notes in Computer Science 2740, 232-246, 2002.
23. David P.: A Constraint-based Approach for Examination Timetabling using Local Repair Techniques. In: E.K. Burke and M.W. Carter (eds): Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference. Lecture Notes in Computer Science, vol. 1408. 169-186, 1998.

24. de Werra D.: An Introduction to Timetabling. *European Journal of Operational Research*, **19**: 151-162, 1985.
25. Di Gaspero L. and Schaerf A.: Tabu Search Techniques for Examination Timetabling. In: Burke E.K. and Erben W. (eds.): Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference (PATAT00), Lecture Notes in Computer Science 2079, 104-117, 2000.
26. Dowsland K.A. and Thompson J.: Ant Colony Optimization for the Examination Scheduling Problem. *Journal of Operational Research Society*, **56**: 426-438, 2005.
27. Glover F. and Kochenberger G.: *Handbook of Metaheuristics*, 457-474, 2003.
28. Glover F. and Laguna M.: *Tabu Search*. Kluwer, Boston, 1997.
29. Joslin D.E. and Clements D.P.: "Squeaky Wheel" Optimization. *Journal of AI Research*, **10**: 353-373, 1999.
30. Karp R.M. Reducibility among Combinatorial Problems. *Complexity of Computer Computations*, 85-103, 1972.
31. Le Huédé F., Grabisch M., Labreuche C. and Savéant P.: MCSA New Algorithm for Multicriteria Optimisation in Constraint Programming. *Annals of Operational Research*, **147**: 143-174, 2006.
32. Lewis R.: A Survey of Metaheuristic-based Techniques for University Timetabling Problems. *OR Spectrum*, **30**(1): 167-190, 2008.
33. McCollum B.G.C.: A Perspective on Bridging the Gap between Theory and Practice in University Timetabling. In: E.K. Burke and H. Rudova (eds.): Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference. Lecture Notes in Computer Science, vol. 3867, 3-23, 2007.
34. Merlot L., Boland N. Hughes B. and Stuckey P.: A Hybrid Algorithm for the Examination Timetabling Problem. In: Burke E.K. and Causmaecker P. (eds.): Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference (PATAT02), Lecture Notes in Computer Science 2740, 207-231, 2003.
35. Meyers C. and Orlin J. B.: Very Large-Scale Neighbourhood Search Techniques. In: E.K. Burke and H. Rudova (eds.): Practice and Theory of Automated Timetabling VI: Selected Papers from the 6th International Conference. Lecture Notes in Computer Science, vol. 3867, 24-39, 2007.
36. Ozcan E., Bilgin N. and Korkmaz E.E.: Hill Climbers and Mutational Heuristics in Hyperheuristics. In: Proc. of the 9th International Conference on Parallel Problem Solving from Nature, 202-211, 2006.
37. Petrovic S. and Burke E.K.: University Timetabling. In: J. Leung (ed.): *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapter 45, CRC Press, April 2004.
38. Qu R. and Burke E.K.: Adaptive Decomposition and Construction for Examination Timetabling Problems. Multidisciplinary International Scheduling: Theory and Applications Conference, 418-425, Aug, 2007, Paris, France.
39. Qu R. and Burke E.K.: Hybridisations within a Graph Based Hyper-heuristic Framework for University Timetabling Problems. Accepted by *Journal of Operational Research Society*, 2008.
40. Qu R., Burke E.K., McCollum B. Merlot L.T.G. and Lee S.Y.: A Survey of Search Methodologies and Automated System Development for Examination Timetabling. To appear in *Journal of Scheduling*, 2009, DOI: 10.1007/s10951-008-0077-5.
41. Rattadilok P. and Kwan R.S.: Dynamically Configured λ-opt Heuristics for Bus Scheduling. In: Burke E.K. and Rudova H. (eds.): Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, 473-477, 2006.
42. Salavatipour M.R. On Sum Coloring of Graphs. *Discrete Applied Mathematics*, **127**(3): 477-488, 2003.
43. Schaerf A.: A Survey of Automated Timetabling. *Artificial Intelligence Review*. **13**(2): 87-127, 1999.
44. Thompson J. and Dowsland K.: A Robust Simulated Annealing Based Examination Timetabling System. *Computers & Operations Research*, **25**: 637-648, 1998.
45. Wren A.: Scheduling, Timetabling and Rostering - A Special Relationship? In: E.K. Burke and P. Ross (eds): Practice and Theory of Automated Timetabling I: Selected Papers from the 1st International Conference. Lecture Notes in Computer Science, vol. 1153. 46-75, 1996.

Appendix A. Trends of the Best LWD Hybridisations with SD for Benchmark Exam Timetabling Problems



Appendix B. Trends of the Best LWD Hybridisations with SD for Benchmark Graph Colouring Problems