Elsevier Editorial System(tm) for European Journal of Operational Research
Manuscript Draft

Manuscript Number: EJOR-D-06-00943R1

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

Article Type: Regular Paper

Section/Category: Timetabling

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

Corresponding Author: Dr. Rong Qu, PhD

Corresponding Author's Institution: School of CSiT, University of Nottingham

First Author: Rong Qu, PhD

Order of Authors: Rong Qu, PhD; Edmund K Burke; Barry McCollum

Manuscript Region of Origin:

Abstract: In this paper we present a random iterative graph based hyper-heuristic to produce a collection of heuristic sequences that consist of different graph colouring heuristics to construct solutions of different quality. These heuristic sequences can be seen as dynamic hybridisations of different heuristics to construct solutions step by step. Based on these sequences, we 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 effort on hybridising Largest Weighted Degree with Saturation Degree at the early stage of solution construction tends to generate good 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.

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 that consist of different graph colouring heuristics to construct solutions of different quality. These heuristic sequences can be seen as dynamic hybridisations of different heuristics to construct solutions step by step. Based on these sequences, we 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 effort on hybridising Largest Weighted Degree with Saturation Degree at the early stage of solution construction tends to generate good 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 heuristics, hybridisation, hyper-heuristic

1 Introduction

Since the 1960s exam timetabling has been one of the most studied subjects in timetabling research (see [18,29]). 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. [8,10,18,29,32]).

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 satisfying 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 the most important problems in graph theory and are known as NP-hard [24].

Graph colouring heuristics such as that in [4] were widely studied in early timetabling research [10,19], and are still being employed nowadays as either the initialisation method for meta-heuristics, or they are being integrated with meta-heuristics in different ways (e.g. [2,6,12,15,16,27]). Meta-heuristics [23] have received significant attention in the last two decades and have been very successful over a range of complex timetabling problems [29]. These include Tabu Search (e.g. [21]), Simulated Annealing (e.g. [4,24,33] and Evolutionary Algorithms (e.g. [13]), etc.

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. [1,27]) by employing different neighbourhood structures

during the search. Iterative techniques such as GRASP have also obtained some success on exam timetabling (e.g. [20,27]). Other new technologies investigated include Case-Based Reasoning e.g. [11,16], fuzzy reasoning (e.g. [2]) and hybrid approaches (e.g. [6,14,17,25]).

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. [3,9,16,26,30]) and constructive strategies (e.g. [6,12,16,27]). Graph heuristics are the mostly 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 [15], 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 adaptively, but not directly, adjusted. It is, rather, made by adaptively calling different ordering strategies in graph heuristics at a higher level. 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 [19], and are publicly available at ftp://ftp.mie.utoronto.ca/pub/carter/testprob/. 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 in [29]. We used version *I* of the data [29] and present the characteristics of these problems in Table 1. If we define a conflict matrix C, where c_{ij}=1 if exams *i* and *j* conflict, c_{ij}=0 otherwise, the *conflict density* in Table 1 can be defined as the density of element c_{ij} of value 1 in the matrix. During the years, variants *a* (exam timetabling) and *b* (graph colouring) of these benchmarks have been tested in the literature. For more details see [29] and http://www.asap.cs.nott.ac.uk/resources/data.shtml.

Table 1 Characteristics of the benchmark timetabling problems in [19], also see [29]

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta92 I	yor83 I
No. of exams	682	543	190	81	461	381	139	261	184	622	181
No. of timeslots	35	32	24	18	20	18	13	23	10	35	21
No. of students	16925	18419	1125	2823	5349	2726	611	4360	2750	21266	941
Conflict density	0.13	0.14	0.27	0.42	0.6	0.6	0.14	0.18	0.8	0.13	0.29

In variant *b* of the problem, a set of (81-682) exams need to be scheduled into a limited number of (10-35) timeslots in different instances. Hard constraints are to avoid students taking two exams at the same time. That is $t_i \neq t_j$; $i \neq j$ and $D_{ij} > 0$ (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.

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

where

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

$w_i = 2^i$ is the cost of assigning two conflicted exams e_l and e_k with i timeslots apart, if $i = |t_l - t_k| < 5$; t_l and t_k as the timeslots of exam e_l and e_k, respectively

e 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 [19]

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 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 enrols the event has
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 [12], heuristic sequences consisting of the five graph heuristics presented in Table 2 and a random ordering strategy were searched by a standard Tabu Search and were used to construct solutions for exam and course timetabling problems. At each step of 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 Tabu Search. Figure 1 presents the pseudo-code of this graph based hyper-heuristic (GHH).

The solution construction at each step of Tabu Search, using a heuristic sequence (of length e, e as 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 //a move in Tabu Search
   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
         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 [12]

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.

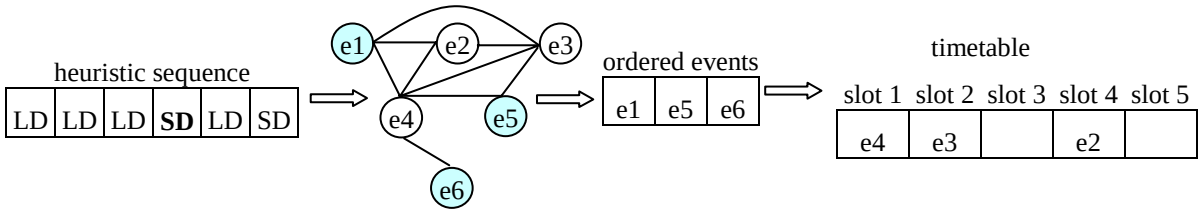


Figure 2 An illustrative example of solution construction using a sequence of graph

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 [12], 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 Fred Glover as "a master strategy that guides and modifies other heuristics" [22]. The *heuristic* here usually refers to the mechanism of moves "for transforming one *solution* into another" [22], rather than moving between heuristic sequences. Most meta-heuristics in the literature operate directly on a search space of solutions but a hyper-heuristic operate 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. In this paper, we develop an adaptive approach where heuristics are dynamically hybridised during solution construction. It is based on the observations upon a large number of different heuristic sequences obtained by a random iterative GHH. The above same solution construction process 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 is developed in Section 5.

```
for i = 0 to i = e × 50, e: the number of exams/vertices
   initialise heuristic sequence h = {SD SD … SD SD}
   for n = 1 to n = e
      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 [15,19] and in our previous work [12] 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. 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, respectively. CD in Table 2 was found not as effective as other heuristics and also time consuming [12], thus is not considered here.

To guarantee a full coverage of different percentages of hybridisation, the random iterative GHH systematically hybridises n LWD, LE or LD, $n \in \{1,\ldots,e\}$, in the sequences. For each percentage of hybridisations 50 samples are obtained. At the first iteration of solution construction, SD always returns the same level of difficulty (number of valid timeslots/colours) for all exams/vertices. So, at the start, LD, LE or LWD will be employed i.e. the heuristic sequences always start 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 percentages of LWD, LE or LD. 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 this sequence generates a solution of penalty 11 using the evaluation function in the problem. 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 between 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 probability of [0,1] 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. early stage of solution construction) tends to generate the best solutions.

III. Finally, lines that show the trends of LD, LE or LWD hybridisations in heuristic sequences are plotted based on the appearances of LD, LWD or LE at different positions of solution construction (of the three subsets of different quality).

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 Percentage	0.8	1	0.2	…	…	…	…	0.2		

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

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 LD, LE or LWD obtained by the random iterative GHH (RGH) for benchmark exam timetabling problems. Best results are in bold. "% of Lx" gives the average overall amount of LWD, LD or LE hybridised in heuristic sequences.

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta92 I	yor83 I
RGH-LE best	5.56	4.75	41.06	13.33	16.05	12.2	165.53	9.48	29.38	4.18	44.45
% of LE	29	11	14	34	24	10	26	15	12	18	13
RGH-LE worst	6.85	5.96	52.66	21.63	21.76	17.14	184.93	11.83	38.56	5.29	51.67
% of LE	27	32	32	37	37	44	36	26	45	28	27
RGH-LD best	5.43	4.47	40.24	12.44	16.06	12.41	163.18	8.87	29.88	4.13	**41.72**
% of LD	29	34	38	26	40	42	10	30	11	26	10
RGH-LD worst	6.39	5.77	49.86	16.74	21.01	16.59	185.02	11.63	38.6	5.37	48.81
% of LD	23	30	14	35	43	45	39	30	24	15	23
RGH-LWD best	**5.26**	**4.43**	**37.95**	**12.15**	**15.37**	**12.01**	**159.58**	**8.76**	**28.98**	**3.95**	42
% of LWD	36	29	47	53	36	23	34	34	63	29	27
RGH-LWD worst	6.06	5.2	49.07	15.28	20.27	15.23	180.47	11.09	34.38	4.95	48.15
% of LWD	25	18	19	9	10	12	31	19	10	17	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 two benchmark problem instances "hec92 I" and "ute92 I" in Table 3. The trends of hybridisations 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).

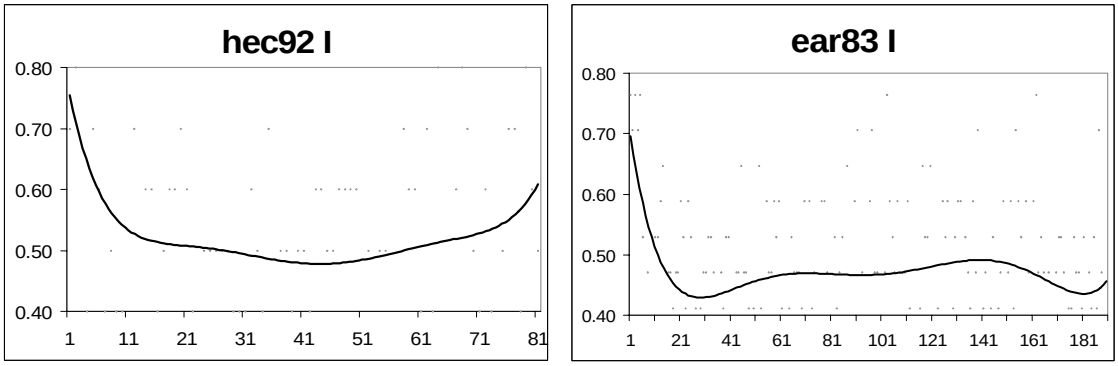


Figure 5 The trends of hybridising LWD in the best heuristic sequences for "hec92 I" and "ear83 I"

Another observation from Table 3 is that the overall amount of LWD hybridisations in the best heuristic sequences for different problems 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 problems such as "sta83 I" and "ute92 I", the range of LWD hybridisation is wider than other problems such as "yor83 I" and "car91 I". This indicates that hybridising LWD for the latter problems may be more difficult within the GHH approach.

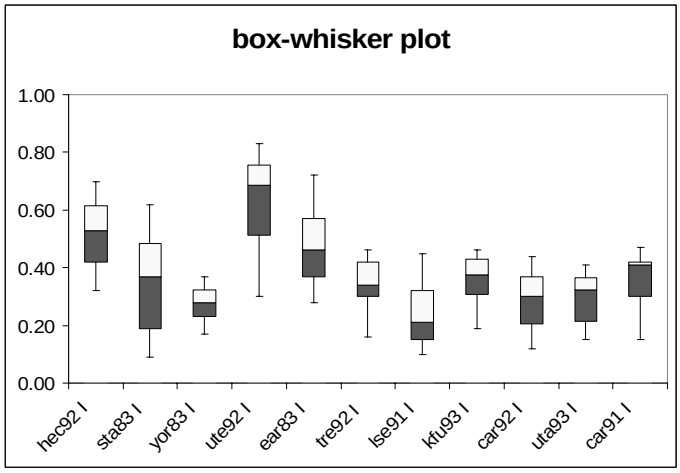


Figure 6 The amount of the best LWD hybridisation in heuristic sequences for exam timetabling problems

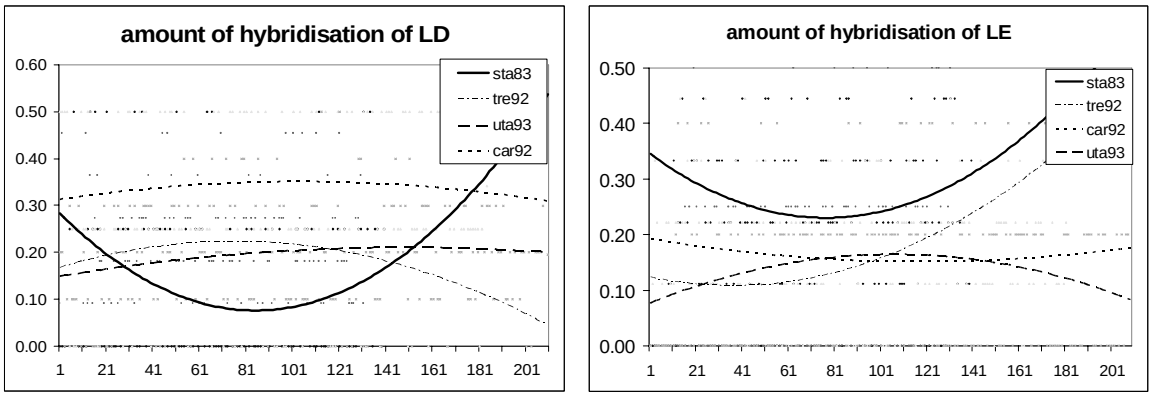


Figure 7 Hybridisation trends of LD and LE in the best 5% heuristic sequences for exam timetabling problems

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 problems and thus no obvious observations are obtained. Due to the space limitation, and also because we are not particularly interested in the characteristics of the worst sequences, we do not present these trends in the paper.

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 random GHH in Figure 3 searches for heuristic sequences.

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 [31] 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. number of colours) are saved for further analysis. Table 4 presents the best and worst results obtained by these sequences for the graph colouring problems. Again we present trends of LWD appearance in the best heuristic sequences for two problems in Figure 8, and the rest of the problems in Appendix B.

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

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta92 I	yor83 I
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 problems. For almost all problems 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 "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.

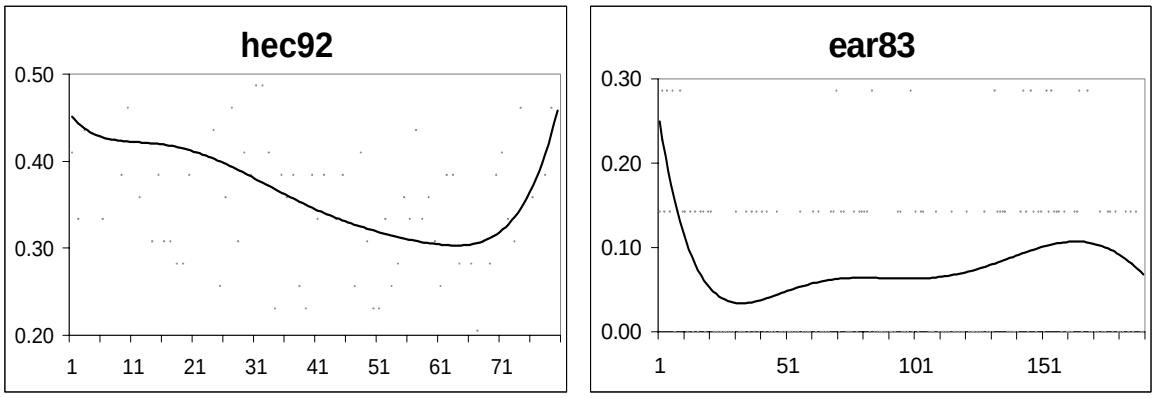


Figure 8 Trends of hybridising LWD in the best heuristic sequences for graph colouring problems "hec92" and "ear83"

The range of the overall amount of LWD hybridisations in Figure 9 again shows that for different problems, the percentages of hybridisation in the best heuristic sequences are very different. The range of hybridisation percentages 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.

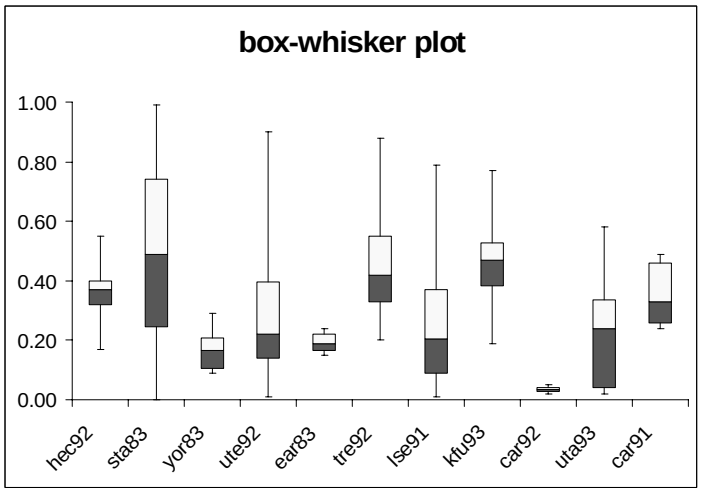


Figure 9 The amount of the LWD hybridisations in the best heuristic sequences for graph colouring problems

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 amount of hybridisations and their appropriate ranges vary a lot for different problems. 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 in the second half of heuristic sequences). This is an adaptive process that adjusts the amount of LWD hybridisations 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 spent on 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 when 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 percentage of LWD to be hybridised in the first half of the sequence is increased by 0.03. The aim is to further explore different hybridisations and avoid converging quickly to a fixed rate. The best heuristic sequences and the best rate are saved and updated in stage II of the adaptive approach. The hybridisation rate is limited within the range [0.1, 0.7], and will be reset as the current best rate if it reaches one end. Note that a hybridisation rate 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 rate is increased by 0.03.

iii) If a feasible but worse solution is generated, the hybridisation rate is decreased by 0.01.

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

5.1 Adaptive Heuristic Hybridisation for Benchmark Exam Timetabling Problems

We test this adaptive approach on the benchmark exam timetabling problems and present the results in Table 5. The average computational time across the 11 instances is also presented for six runs on a Pentium IV machine with 1GB memory. The number of iterations is $e/3$ in stage I and $e\times2$ in stage II for small problems, and $e/5$ in stage I and e in stage II for large problems, respectively. We also implemented an iterative hybrid approach with a fixed hybridisation of 20% LWD to SD for $e\times50$ iterations. For easy comparison, the best results from the random iterative GHH (as shown in Table 3) are also presented in Table 5.

Table 5 Results from the adaptive approach (AGH), fixed hybridisation (RGH 20%), and random hybridisation of LWD (RGH). Computational time is presented in seconds, and the best results are in bold.

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta92 I	yor83 I
AGH average	5.2	4.39	37.67	12.16	15.47	11.57	160.1	8.78	29.07	3.36	42.6
% of LWD	27	25	37	40	30	20	19	26	30	29	47
AGH best	**5.17**	**4.32**	**35.7**	**11.93**	**15.34**	**11.45**	**159.05**	**8.68**	**28.88**	**3.3**	**40.79**
Time (s)	13247	4883	252	14	602	905	56	732	59	7710	219
RGH 20% best	5.2	4.4	49.74	12.53	15.6	11.52	**159.05**	8.89	29.36	3.6	41.77
Time (s)	38078	12600	2777	289	10037	6007	797	5258	505	21948	2463
RGH best	5.26	4.43	37.95	12.15	15.37	12.01	159.58	8.76	28.98	3.95	42
% of LWD	36	29	47	53	36	23	34	34	63	29	27
Time (s)	31115	11687	2671	179	9452	5710	450	5012	503	20561	2381

The adaptive approach obtained the best results on all problems. Comparisons with "RGH 20%" indicate that hybridising LWD adaptively rather than at a fixed rate contributes to a better performance. Note that "RGH 20%" requires a much larger computational time. For different problems, the levels of LWD hybridised in the best sequences belong to the range [19%, 47%].

Compared with the random iterative GHH, the adaptive approach is more effective on all problems in a much quicker time. The amount of LWD hybridisation obtained by the adaptive approach is within the range found by the random iterative GHH. 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.

We also compare this adaptive approach with those in the literature in Table 6, 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 literature was not reported.

Table 6 Best results from the adaptive approach (AGH), GHH using Tabu Search (TS-GHH), and other approaches in the literature on the benchmark exam timetabling problems.

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta92 I	yor83 I
AGH	*5.17*	*4.32*	*35.7*	*11.93*	*15.3*	11.45	159.05	*8.68*	*28*	*3.3*	*40.79*
TS-GHH [12]	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**
[2] (2004)	5.29	4.56	37.02	11.78	15.81	12.09	160.42	8.67	27.78	3.57	40.66
[5] (2004)	4.8	4.2	35.4	10.8	13.7	10.4	159.1	8.3	25.7	3.4	36.7
[14] (2003)	**4.65**	**4.1**	37.05	11.54	13.9	10.82	168.73	8.35	25.83	**3.2**	37.28
[15] (2004)	4.97	4.32	36.16	11.61	15.02	10.96	161.91	8.38	27.41	3.36	40.77
[17] (2001)	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	158.2	9.4	24.4	3.5	36.2
[19] (1996)	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	25.8	3.5	41.7
[21] (2000)	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	29.0	4.2	42.0
[25] (2003)	5.1	4.3	35.1	10.6	13.5	10.5	157.3	8.4	25.1	3.5	37.4
[27] (2007)	5.45	4.5	36.15	11.38	14.74	10.85	**157.2**	8.79	26.68	3.55	42.2

We can also observe from Table 6 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 outperforming the others. Note that most of the other approaches are specially designed for the particular problems, and require initial solutions.

The adaptive hybrid approach is an efficient and much simpler method compared with the previous GHH using Tabu Search [12], which required much larger computational time. Not only a larger number of iterations, but also a steepest descent were needed to further improve each compete solution. This added much more computational expense. The adaptive approach obtained better results (in italics in Table 4) with a much shorter computational time compared with the previous TS-GHH approach.

5.2 Adaptive Heuristic Hybridisation for Graph Colouring Problems

We test the same adaptive approach also on the benchmark graph colouring problems to demonstrate the generality of this method. The results are presented in Table 7. We found that the approach quickly obtained the same or better results compared with the random iterative GHH in a much shorter time even without further adjustment based on the sequences obtained by the adaptive hybridisation on the first half of the sequences. Thus the stopping condition of the random iterative GHH and adaptive approach is set as when the best results are obtained to further demonstrate the efficiency of the adaptive approach. The computational time is presented in Table 7.

Table 7 Best results from the adaptive approach (AGH) and random iterative GHH (RGH) on graph colouring problems. The best results reported in the literature are also presented.

	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 [29]	**28**	**28**	**22**	**17**	**19**	**17**	**13**	**20**	**10**	**30**	**19**

We can see that for 3 problems, the two approaches take the same time to obtain the best results. For the rest of the problems, the random iterative GHH needs much more time to obtain the same result or a worse result compared with the adaptive approach. The percentage of the LWD hybridisation is also presented in Table 7, 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.

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 both benchmark problems. Note that most of the approaches were specifically developed by humans rather than automatically generated (as is the case here).

The adaptive approach is developed based on the analysis on a collection of heuristic sequences of differing quality. They are obtained by using a random iterative graph based hyper-heuristic 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 solutions for different problems, but also the obtained heuristic sequences can be further analysed for in-depth insight of heuristic hybridisations. Our observations indicate 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 rate is different for different problems (and that it also varies a lot at the early stage of solution construction).

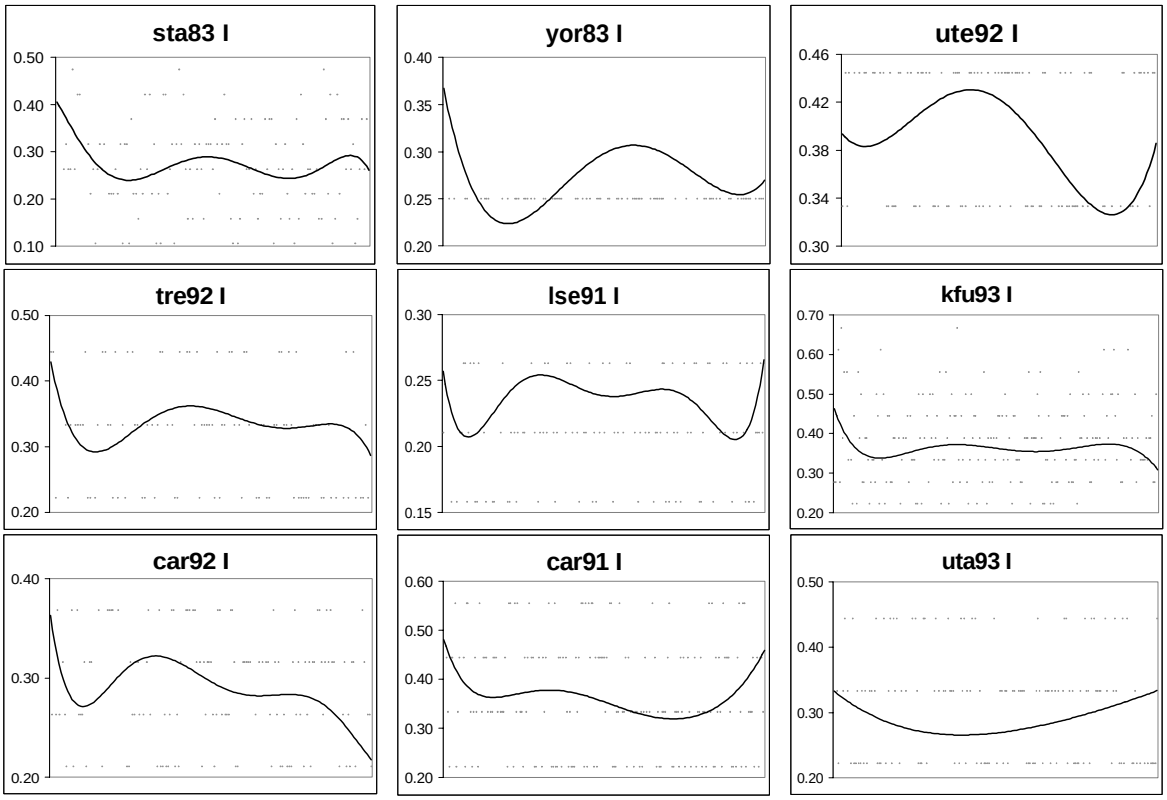
In terms of future research directions, it is interesting to note that as the solutions generated by different heuristic sequences correspond to very different solutions in the solution space [12], it will be beneficial to investigate this approach on generating diversified initial solutions for different 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

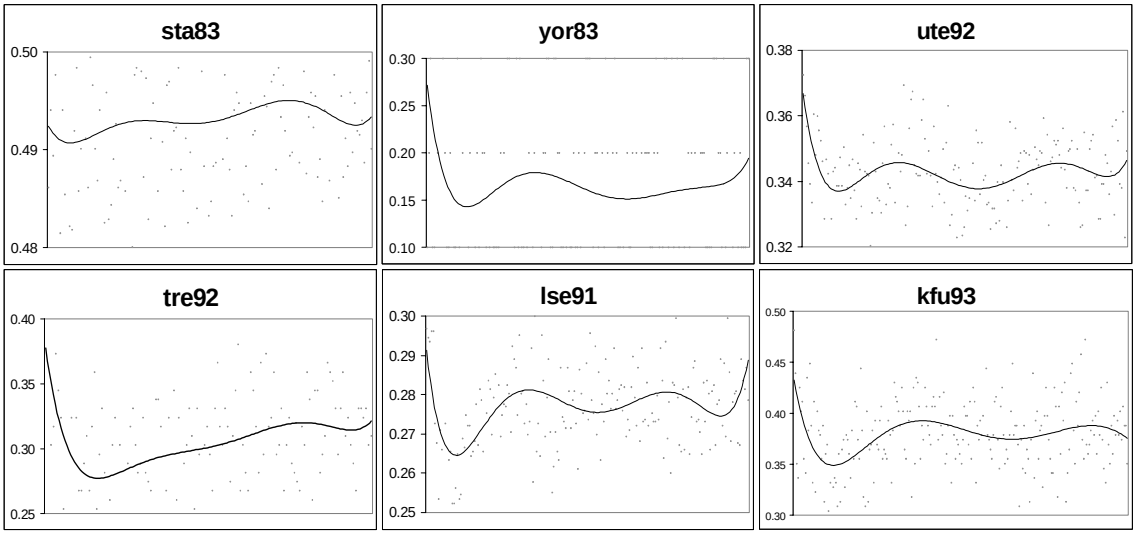
1. Abdullah S., Ahmadi S., Burke E. and Dror M.: Investigating Ahuja-Orlin's Large Neighbourhood Search for Examination Timetabling. Accepted by *OR Spectrum*, 2007.
2. Asmuni H., Burke E.K. and Garibaldi J.: Fuzzy Multiple Ordering Criteria for Examination Timetabling. In: Burke E. and Trick M. (eds.): Practice and Theory of Automated Timetabling: Selected Papers from the 5th International Conference, Lecture Notes in Computer Science 3616, 334-353, 2004.
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., Kendall G. and Soubeiga E.: A Tabu Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*. **9**(6): 451-470, 2003.

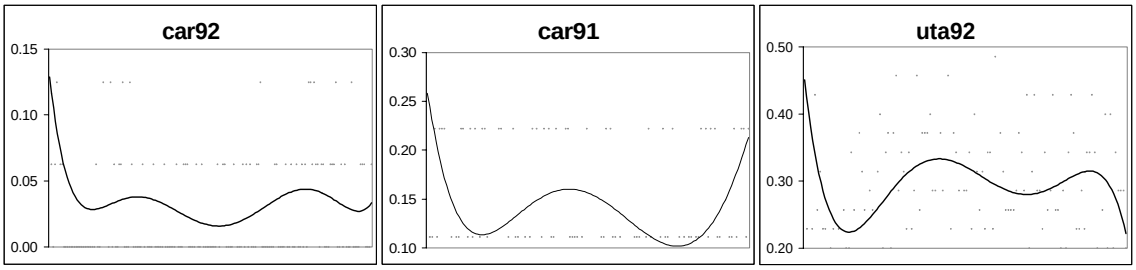
10. Burke E.K., Kingston J. and de Werra D.: Applications to Timetabling. In: Gross J. and Yellen J. (eds.), Handbook of Graph Theory. Chapman Hall/CRC Press. 445-474, 2004.
11. Burke E.K., MacCarthy B., Petrovic S. and Qu R.: Multiple-Retrieval Case-Based Reasoning for Course Timetabling Problems. *Journal of Operational Research Society*, **57**(2): 148-162, 2006.
12. 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, 2006.
13. 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.
14. 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.
15. Burke E.K. and Newall J.P.: Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings. *Annals of operations Research*, **129**: 107-134, 2004.
16. Burke E.K., Petrovic S. and Qu R.: Case Based Heuristic Selection for Examination Timetabling. *Journal of Scheduling*, **9**(2): 99-113, 2006.
17. Caramia M., DellOlmo P. and Italiano G.: New Algorithms for Examination Timetabling. In: Naher, S., Wagner, D. (eds.) Algorithm Engineering 4th International Workshop, WAE 2000. Lecture Notes in Computer Science 1982, 230-241, 2001.
18. 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.
19. Carter M., Laporte G. and Lee S.: Examination Timetabling: Algorithmic Strategies and Applications. *Journal of Operational Research Society*, **47**: 373-383, 1996.
20. 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.
21. 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.
22. Glover F: Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Researc*h. **13**(5): 533 – 549, 1986.
23. Glover F. and Kochenberger G.: *Handbook of Metaheuristics*, 457-474, 2003.
24. Karp R.M. Reducibility among Combinatorial Problems. *Complexity of Computer Computations*, 85-103, 1972.
25. 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.
26. 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.
27. Qu R. and Burke E.K.: Adaptive Decomposition and Construction for Examination Timetabling Problems. To appear at Multidisciplinary International Scheduling: Theory and Applications Conference, Aug, 2007, Paris, France.
28. Qu R. and Burke E.K.: Hybridisations within a Graph Based Hyper-heuristic Framework for University Timetabling Problems. Technical Report NOTTCS-TR-2006-1, School of CSiT, University of Nottingham. 2006. Submitted to *JORS*, 2007.
29. 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. Technical Report NOTTCS-TR-2006-4, School of CSiT, University of Nottingham. Accepted by *Journal of Scheduling*, 2007.
30. 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.
31. Salavatipour M.R. On Sum Coloring of Graphs. *Discrete Applied Mathematics*, **127**(3): 477-488, 2003.
32. Schaerf A.: A Survey of Automated Timetabling. *Artificial Intelligence Review*. **13**(2): 87-127, 1999.
33. Thompson J. and Dowsland K.: A Robust Simulated Annealing Based Examination Timetabling System. *Computer & Operations Research*, **25**: 637-648, 1998.

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




car92
0.15
0.10
0.05
0.00
car91
0.30
0.25
0.20
0.15
0.10
uta92
0.50
0.40
0.30
0.20

Summary of changes made to address the referees' comments

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

R. Qu, E. K. Burke and B. McCollum

(Previously "Adaptive Automated Construction of Hybrid Heuristics for Exam Timetabling Problems")

We thank the referees for their supportive comments and suggestions. They have helped us to improve the paper. We have significantly revised the paper to address the concerns of the three referees and we outline the changes that we have made in this document. In particular, we have added another set of testing problems (graph colouring instances) with further analysis on heuristic hybridisations. The adaptive approach is further improved in the revised version with supporting experimental results. Additional sections (see below) are also added in the modified paper to address the referees' concerns.

Response to Referee 1:

This is an interesting paper on hybrid heuristic construction for exam timetabling problems. This referee recommendation is to require that the authors prepare a revised version in order to improve the organization of the paper and to reduce its length to a short communication.

We have tried to reduce certain parts of the original paper to improve the organisation and to keep the paper as short as necessary. However, we have had to add some sections to address the other referee's concerns. We have moved the problem descriptions forward in Section 2 to underpin a better understanding of the GHH approach. However, to address the valid comments from another referee we have added more details of the analysis and the adaptive approach. Furthermore, we have added more experiments on another set of problems (graph colouring problems) and we have included an appendix of all the heuristic trends to further demonstrate the adaptive approach that we have developed. Also an illustrative example is added to Section 3 and pseudo-code is added to improve the presentation of the paper (also required by the other referee). The modified paper now contains 15 pages. We believe that these amendments have greatly improved the paper.

* The major concern of this referee is related to the organization of the paper. The notion of hyper-heuristics is clearly defined in the introduction, but things become confusing when the authors apply the notion to deal with the exam timetabling problem. More specifically, the authors should specify clearly the set of low level heuristics. Are they the graph heuristic LD, LWD, LE, and SD? If so, how is characterized the upper level heuristic choosing the low level heuristics?

We have reorganised the paper by describing the exam and graph colouring problems in Section 2, and including only the previous work in Section 3. We also added more details of the previous GHH approach (i.e. how Tabu Search is used to search on heuristic sequences) and clearly stated which low level heuristics are included in the previous and current work. An illustrative example is also added to clearly present how low level heuristic sequences are used to construct solutions in the hyper-heuristics.

In Section 2, the procedure summarized in fig.1 is confusing unless the sequence of low level heuristics used at each iteration is fixed a priori. Also, it is difficult to understand the paragraph: " For some heuristic permutations, it permutation is evaluated."

We have specified the length of the heuristic sequences. The paragraph mentioned here, which is about the high level Tabu Search, has been re-written.

In summary, the authors should state clearly in a few words the GHH in the context of the current problem (exam timetabling). Then afterward, they could make the connection with other works. It is difficult for the reader to easily understand the different variants (random iterative GHH and the adaptive version) since the basic GHH is not clearly specified. The reader gets lost in all kinds of considerations instead of being introduced to the processes used.

As Section 2.1 is largely re-organised with more details on Tabu Search, we believe that these issues are clarified now. Furthermore, the motivation for using GHH at a high level to hybridise the low level graph heuristics are discussed to place our current work in a broader context.

* Relevant information related to solution time and number of iterations used is mentioned in different parts of the paper. This information should be regrouped in the section related to numerical results instead. This would be useful to neatly compare the different variants.

We have now placed all this information on experimental settings together in Section 4.1.

* In Section 4, the formula (1) is not clear. The summation is done over what index? The notion of wi should be clarified.

We have added more details to define more formally the evaluation function (now Equation 1). Also, some issues leading to confusion about different problem datasets are also addressed.

* The length of the paper is excessive in view of the scientific content since the notion of GHH has already been introduced.
The paper should be reduced to a short note clearly summarizing how GHH is applied as an adaptive process to deal with the timetabling problem.

As stated above, to clearly explain the work presented in this paper and also to address the other refereeing comments, we have added more details and more scientific content in the paper. Furthermore, more analysis and experiments are added to further analyse the heuristic sequences and the adaptive approach on a benchmark graph colouring problem. The extended paper contains more content.

Response to Referee 2:

Summary: In this paper a random iterative graph (coloring) hyper-heuristic is presented. A collection of heuristics, which are widely used in solving timetabling problems, generates diverse solutions.

Remarks:

1) The subject of the paper is very interesting and relevant.

2) Fig. 1 shows the way a basic heuristic woks, but something is missing to explain a hyper-heuristic.

We have re-organised and re-written a large part of Section 2.1 (now Section 3) to better explain the previous GHH approach. More details about the high level Tabu Search are added. The solution construction process is further explained with pseudo-code and an illustrative example.

3) In my opinion, specific examples would make the paper more complete / attractive. For instance, in table 1, the heuristics could be detailed using a colored graph.

We have extended and improved the original Section 2.1 (now Section 3), and have added more details including an illustrative example in Figure 2.

4) The HGH-LWD can provide good quality solutions, however these don't attain the best solutions in any instance. In this paper this could be explained, since

the hybrid heuristics combine advantages of a collection of heuristics. So, we would expect better computational results.

The goal here is to automate the heuristic design process. We are comparing the computer generated methodology with highly sophisticated human systems. In these circumstances we would not expect to be beating these approaches but argue that to be competitive with them is a publishable scientific achievement. We have explained this now in the paper.

5) Page 11 line -3: In my opinion data mining techniques seem to be out of context. What kind of techniques is the author referring to?

At this stage, this part of our plan for future work is not quite clear yet. Thus, we have removed the text relating to data mining techniques from the future directions section.

6) Page 11 paragraph -2: The generation of different initial solutions is elementary in hybrid heuristic. I think the author must develop this matter in the paper, in other to get better results.

We do agree with the referee that we could develop more algorithms with the adaptive approach here as the initialisation methods. This work requires more systematic experiments and investigations on different hybridisations of hyper-heuristics and other search algorithms such as meta-heuristics. As the improved adaptive approach on its own outperformed the previous Tabu Search hyper-heuristics hybridised with greedy local search, we have indicated this as a direction of future work in the conclusion section. The aim of this work is to develop a simple, fast and general adaptive approach which is capable of intelligently building heuristics. We have demonstrated the effectiveness and the genericity on both exam timetabling and graph colouring problems.

7) Although in the computational results the quality of the solutions is presented, the computational times are not given. In order to compare with other approaches, it would be interesting to present this omitted data.

To give an indication of the computational time for the approaches developed in this paper, we added the computational time and specification of the machine used in the experiments.
The reason we didn't include the computational time in the previous paper is due to the lack of information of computational time in most of the currently reported approaches in the literature. This is mainly because the computational time is not comparable across different platforms. So we only gave the number of iterations in the previous paper. We added these reasons in the resubmission.

Recommendation:
The paper could be improved in the description of the algorithms and in the computational results.

We believe that we have improved the description of the algorithms, and added more details of the computational results.

Response to Referee 3:

The paper contains an interesting investigation on the automatic construction of sequential graph-coloring heuristics applied to the examination timetabling problem. The algorithm proposed comes from an analysis of the hybrid sequential heuristics on real-world benchmarks.

Unfortuntately, the paper lacks in several aspects, which are detailed in the following.

First, looking at the experimental part, some of the presented results (Table 4) seem not so meaningful to me. An average on three runs is absolutely not enough if the algorithm is stochastic in nature.

We have implemented more experiments (in total six runs for each test) to have a better average indication of the updated results in Table 3. To clearly present the analysis, we added extra figures showing the trends of heuristic hybridisations for two example problems (with Appendices for the rest of the problems). Also box-whisker plots are provided to give a more in-depth analysis of the heuristic hybridisations.

Furthermore, drawing conclusions on the basis of the minimum values found is not a statistically sound procedure.

As explained above, in the revised paper we have implemented more experiments and presented more details on the analysis, with best and worst, results, computational time and box-whisker plots.

Moreover, there is no mention on the running times of the algorithms and no report about the experimental settings. Also these points should be described to help the reader understand the practical relevance of your work.

We have added more details of computational time, and explained why comparisons of computational time are impossible because other approaches in the literature did not present the computational time in their work.

Moving to comments on the paper, let's start with the keywords: the term hyper-heuristics (and all the discussion thereafter) is in my opinion misleading. What you describe is a way to hybridise sequential heuristics that DEEPLY depend on the problem at hand; therefore in this case there is no clear connection with what the authors call hyper-heuristics, because, as the authors advocate, hyper-heuristics do not depend on the problem specific details and here the authors deal with GraphColoring heuristics. Moreover, the amount of problem specific information incorporated in metaheuristics is more or less the same of what you call hyper-heuristics, the only difference is that what you call hyper-heuristics work on a (problem-specific) indirect (procedural) encoding. Concerning this point, I suggest to remove that keyword and review the introduction according to these comments.

This confusion is partially because the context of the previous Tabu Search hyper-heuristic was not presented clearly. We have added more details about the previous work on graph based hyper-heuristics in Section 3. The rest of the paper is more concentrated with the idea of automatically hybridising graph heuristics at a higher level.
We have clarified the difference between meta-heuristics and hyper-heuristics at the end of Section 3. We also presented the differences between the direct adjustment on the order of exams, and the adaptive adjustment on graph heuristics at a higher level without directly concerning the exams and vertices.
We agree that the sequential heuristics depend on the problem at hand. The generality of the hyper-heuristic refers to the high level search, which does not concern problem specific information but rather a search on low level heuristics which adaptively address the instance at hand. We have added more details in Section 1.
We hope we have clarified the issue the referee concerned here. The key point here is that we are developing an automated approach to *build* heuristics rather than to directly solve the problem. The challenge is to build computer systems which are capable of competing with humans in designing heuristic methods.

Finally, the main work on what you call hyperheuristics is (Glover and Laguna 1997) "A master strategy that guides and modifies other heuristics", that must be cited!

We have cited the relevant reference at the end of Section 3, and explained the difference between the standard implementation of metaheuristics and hyper-heuristics. The main point is that most implementations of metaheuristics operate directly on a search space of potential solutions whereas a hyper-heuristic will operate on a search space of heuristics. Of course, metaheuristics can be employed as hyper-heuristics and we have made this clear in the paper.

Some other punctual comments:
Introduction:
p. 2 line 19 (e.g., [1, 34]) instead of i.e.

This is done.

p. 2 4th paragraph, the discussion of incorporating problem specific information in meta-heuristics should be addressed according to the above comments.

We have clarified this by giving more details in the last paragraphs in Section 3.

p. 3 Fig. 1. First, is not the right place to put the figure (since its discussion is in the next page); Secondly, there is no mention on how the schedule is performed (i.e., the first least-cost timeslot is chosen).
I also do not know whether the figure is so meaningful, perhaps a pseudo-code would be a better way to present the idea.

We have re-organised Section 2.1 (now Section 3) and added more details of how GHH works. Pseudo-code, in conjunction with more details of the method, is provided. In addition, an illurstrative example is presented.

p. 4 2nd paragrah, what you describe has a precise name, which is backtracking!

We have changed this description and added it in the pseudo-code.

p. 4 Sect. 2.2, line 3. The authors should introduce what a heuristic permutation is.

We have changed all occurrences of “heuristic permutation” in the paper to “sequence of graph heuristics” or “heuristic sequences” as the latter are more precise terms.

p. 4 line -8. It is not clear how the heuristic permutations are generated. Since this point is the core of one of the techniques employed it deserves more explanation.

We have changed the description and added some pseudo-code to clearly explain the process.

p. 5 line 3. The term injecting could be presented before.

We have removed this term to avoid confusion. As this text is not crucial to the rest of the paper this will not affect the understanding of the work presented here.

p. 5 Sect. 2.3. Much detail on the settings of the experiment employed in the analysis must be provided. It is not clear how the percentages appear in the following table, how they were chosen, and so on.

We have presented all information about experimental settings together at the beginning of Section 4.1 (as suggested by another referee, see above). Pseudo-code is also provided to provide precise information of the process.

p. 5 lines -5,-3. The part about the reasons for which LWD performs good is not so clear to me.

We re-wrote this and added more details in Section 4.1.

p. 6 The analysis made with Microsoft Excel must be detailed! What kind of analysis did you perform? What were the parameters? A rigorous explanation is needed!
p. 6 point III what's polynomial trendline of MS Excel? Which kind of statistical analysis it performs? The same comment as the previous point applies.

We have removed the text about MS Excel tool to avoid any confusion, as the idea of trend lines are general and can be generated using different tools. A general description instead is given to explain the idea.

p. 8 Sect. 4 line 1 that [were] firstly presented by Carter...

Done.

p. 8 Formula 1: the set of numbers 0, 1, 2, 3, 4 should be in curly parentheses

We have added this and provided a more formal definition of the evaluation function as another referee required.

p. 9 Sect. 4.1 Why do you perform only three runs? You could not draw any sound conclusions on such a small number of samples. In a way you are invalidating all the rest of the analyses. What's the statistical significance of the test you employed (since it is not a standard procedure I am not so sure it is so easy to say something about).

We have added more experiments and updated the results in Table 3. Also, the best and worst results, together with computational time and box-whisker plots, are provided to facilitate a deeper analysis in the revised paper.

p. 11 line 1 when you mention greedy local search do you mean First Descent or Steepest Descent?

We meant steepest descent. However, as the updated adaptive approach further improved the results, the approach on its own outperformed the previous Tabu Search hyper-heuristics hybridised with local search. Thus this simple greedy search is now removed from the paper. More thorough investigation and experiments on efficient hybridisations between meta-heuristics and hyper-heuristics will form one of our future research directions. This is included in the conclusion section.

In several places "which" should be preceded by a comma (or you should use that instead).

We have made these changes.

In several places "= number ..." is absolutely colloquial and must be avoided.

We have removed these by defining *e* as the number of events, and referring to it later in the paper.

In summary:

The paper has been revised thoroughly to include more details not only on the approach, but also on extra benchmark problem data. In particular, we have added more details about the basic GHH approach to clarify some of the issues raised. More experimental analysis is carried out and computational time is addressed.

We thank again the referees for their very constructive comments. We believe that we have addressed their concerns and that, as such, we have significantly improved the paper.