Hybridisations within a Graph Based Hyper-heuristic Framework for University Timetabling Problems

Rong Qu and Edmund K Burke
Automated Scheduling, Optimisation and Planning (ASAP) Group
School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
rxq@cs.nott.ac.uk, ekb@cs.nott.ac.uk

Abstract. A significant body of recent literature has explored various research directions in hyper-heuristics (which can be thought as *heuristics to choose heuristics*). In this paper, we extend our previous work to construct a unified graph based hyper-heuristic (GHH) framework, under which a number of local search based algorithms as the *high level* heuristics are studied to search upon sequences of *low level* graph colouring heuristics. To gain an in-depth understanding on this new framework, we address some fundamental issues concerning neighbourhood structures and characteristics of the two search spaces (namely the search spaces of the heuristics and the actual solutions). Furthermore, we investigate efficient hybridisations in GHH with local search methods and address issues concerning the exploration of the high level search and the exploitation ability of the local search. These, to our knowledge, represent entirely novel directions in hyper-heuristics. The efficient hybrid GHH obtained competitive results compared with the best published results for both benchmark course and exam timetabling problems, demonstrating its efficiency and generality across different problem domains. Possible extensions upon this simple yet general GHH framework are also discussed.

Keywords: university timetabling, graph colouring heuristics, hyper-heuristics, Tabu Search, Variable Neighbourhood Search, Iterated Local Search

1. Introduction

Timetabling Problems

Timetabling problems have been widely investigated in the operational research and artificial intelligence research communities for more than four decades (e.g. see [16, 45, 49, 53]). They appear in the forms of educational timetabling (e.g. [5, 20, 23, 24, 44, 45]), nurse rostering (e.g. [10]), sports timetabling (e.g. [29]) and transportation scheduling (e.g. [37]).

This paper is concerned with university (course and exam) timetabling problems, which have attracted particular attention over the years. They are important administrative activities which have to be addressed on a regular basis in almost all academic institutions. There are several surveys of educational timetabling

research covering a variety of research issues (e.g. [13, 16, 20, 44, 45, 49, 53]). Specific papers on course [24] and exam [23] timetabling provide excellent surveys up to 1998 and 1996, respectively. A discussion of the difference between course and exam timetabling is presented in [4140].

In [16], the goal of a general timetabling problem is defined as being *"to assign times and resources to the meetings so as to satisfy the constraints as far as possible"*. This is to assign a set of events (courses or exams) to a limited timeslots while satisfying a number of constraints. Constraints in university timetabling can usually be grouped into two types:

- *Hard Constraints* which must be satisfied under any circumstances. For example, conflicting events (i.e. those courses/exams/meetings with common resources such as students) cannot be scheduled simultaneously (i.e. $x_i \neq x_j$, $i \neq j$ and $D_{ij} > 0$; D_{ij}: the number of students in both events i and j; x_i: the time period that event i is assigned). Room capacity needs to be large enough to accommodate all students (i.e. $\sum s_i \leq C_t$; $x_i = t$; s_i: number of students taking exam i; C_t: room capacity of time period t). Solutions which satisfy all of the hard constraints are often called *feasible* solutions.

- *Soft Constraints* are desirable but not essential requirements. For example, in exam timetabling, it is preferable to avoid students sitting two exams in consecutive time periods. On the other hand, in course timetabling, students may prefer to have consecutive courses.

Soft constraints are usually used to evaluate how good the feasible timetables are. For real world university timetabling problems, it is usually impossible to find solutions satisfying all the soft constraints. Indeed, it is sometimes very difficult to just find a solution which satisfies all the hard constraints.

Timetabling problems in their simplified form (i.e. with only the conflicting hard constraints) can be modelled as graph colouring problems [16]. Graph colouring heuristics represent some of the very early timetabling approaches (from the 1960s) (e.g. [16, 53]). Constraint based techniques have also been employed (e.g. [39, 54]) by modelling timetabling problems as Constraint Satisfaction Problems [7]. Meta-heuristics [14, 46] have been very successful over a range of complex timetabling problems in recent research. These include Tabu Search (e.g. [31, 43, 55]), Simulated Annealing (e.g. [9, 28, 52]) and Evolutionary Algorithms (e.g. [18, 51]).

Recent new techniques and methodologies on timetabling include Ant Algorithms (e.g. [30, 50]), Case-Based Reasoning (e.g. [11, 21]), fuzzy reasoning (e.g. [4]), GRASP (e.g. [26]) and hybrid approaches [22]. Innovative neighbourhood design in search algorithms has also obtained some success (i.e. Very Large Scale Neighbourhood Search [1, 40], multi-neighbourhood search [3] and Variable Neighbourhood Search [2], etc). An overview of the timetabling literature can be seen by consulting the following papers: [23, 24, 45, 49].

1.2 Hyper-heuristics

Most meta-heuristic [14, 46] implementations need to be fine tuned for solving particular problems, and such fine-tuned methods do not usually transfer easily to different problems. The definition of hyper-heuristics is rather simple. They are "heuristics to choose heuristics" [12, 47]. They search upon a search space of heuristics rather than actual solutions. This implies that *any* heuristics (meta-heuristics and population based algorithms, etc) could be employed as *high level* heuristics to operate upon a search space of *low level* heuristics, which could also, essentially, be any heuristics. In most of the papers in the literature, meta-heuristics are employed directly on a solution based search space. There is, of course, the possibility of using meta-heuristics as the "low level" heuristics (i.e. those to be selected) in a hyper-heuristics approach but, as far as we are aware, this has never been addressed in the literature.

The heuristics being searched are used (at a *lower level*) to solve the actual problems. The *high level* heuristics choose from a set of low level heuristics rather than dealing with the actual elements in the problems. The driving motivation behind this newly emerging framework is to raise the generality of search algorithms in order to solve a wider range of problems. By defining our search spaces upon heuristics rather than actual solutions, hyper-heuristics avoid dealing with the specific details of problem domains. Domain knowledge of the actual problems is left to the low level heuristics. The idea is that the high level search should be more generic. The representation of the search space is usually quite simple (rather than having complex structures representing problem solutions). The role of the high level search is purely to manage and learn upon the set of low level heuristics so that they can be adaptively applied to a range of problems.

The idea of hyper-heuristics originated in 1960s, although, at the time, the particular term was not used [12]. It is thus related to some algorithms, however, is also fundamentally different from most heuristic approaches. For example, in resource constrained project scheduling [8], permutations of jobs are usually obtained by using certain (fixed) strategies and jobs are then scheduled one by one in this specific order. In hyper-heuristics, the aim is to *search* for a *set* of (low level) strategies which permute the jobs *during* the scheduling to construct high quality solutions. The goal is to learn to adaptively adjust the low level strategies used at different stages of the scheduling. In [27], a framework was built to integrate multiple heuristics within a stochastic sampling search algorithm, where online statistical models of the search performance were used to select heuristics during the search for resource-constrained project scheduling. As an emerging research direction, the field of hyper-heuristics is also strongly related to some other research in the literature. In [33], algorithm portfolios were developed where a set of heuristics was used for solving combinatorial optimisation problems. In [42], reinforcement learning was used to adaptively choose promising heuristics during the search process within a constraint programming environment for two optimisation problems.

Hyper-heuristics have been investigated either on using moving strategies (i.e. [15, 28, 32]) or on using constructive heuristics [6, 11, 17, 21, 48, 51] as the low level heuristics. In [15], Tabu Search was employed as the high level heuristic upon a set of moving strategies on both the nurse rostering problems and course timetabling. A Simulated Annealing hyper-heuristic was developed in [28] to deal with the shipper sizes in transportation problems. In [32], a distributed choice function was also proposed as a hyper-heuristic for exam timetabling problems.

Another body of hyper-heuristic work has employed constructive heuristics as the low level heuristics. In [6], heuristic selection and move acceptance criteria were analysed in depth within a hyper-heuristic for both optimisation functions and exam timetabling problems. In [21], graph colouring heuristics were selected by using Case Based Reasoning to order the events upon problem solving situations. A Genetic Algorithm was also developed in [48] to evolve event picking and slot picking heuristics to construct timetables for both class and exam benchmark timetabling problems. Issues of representation and fitness functions were also addressed. Constraint satisfaction strategies were also evolved by using Evolutionary Algorithms in [51] for constructing exam timetables.

In our previous work [17], the search space under consideration consisted of sequences of graph colouring heuristics as the low level heuristics. We employed Tabu Search for solving both course and exam timetabling problems. We also raised the issue of two search spaces in hyper-heuristics. Based on the above approach, in this paper, we formally define a unified graph based hyper-heuristic (GHH) framework, within which a number of high level heuristics are analysed systematically to obtain a deeper understanding of the framework. Important issues on neighbourhood structures and the search space of heuristics, compared with search space of solutions, are addressed. Within the research covered in the hyper-heuristic literature, to our knowledge, there is no work which analyses the structure and nature of the search space of heuristics in this way. Hybridisations of the GHH framework with local search methods within the solution space are also investigated on both course and exam benchmark timetabling problems. Note that exactly the same GHH is applied for both problems which are quite different in terms of problem constraints [41]. There are very few papers in the literature which deal with both problems by using the same system.

1.3 Benchmark Course and Exam Timetabling Problems

The course timetabling problems tested in this paper were generated by the Meta-heuristic Network (http://www.metaheuristics.net/). The problem set consists of 11 problems in three (small, medium and large) sizes, where from 100 to 400 courses need to be assigned into 45 timeslots, 9 timeslots per day for 5 days of a week. Room capacity and features need to be considered to accommodate all the students in a particular course. We list both the hard and soft constraints in Table 1. The penalty of the feasible timetables generated is the sum of the number of violations of the soft constraints.

INSERT TABLE 1 HERE.

The exam timetabling problems we tested in this paper were first introduced in [25], and have been widely tested by a number of approaches during the last ten years [45]. The dataset consists of 13 problems from different institutions, among which 11 have been more heavily investigated because of errors in the other two problems. A more detailed discussion of those datasets (and the difficulties caused by different instances circulating under the same name) was given in [45]. The constraints in the problems can be outlined as follows:

- *Hard constraint*: no conflicting exams (with common students) should be scheduled into the same timeslot.

- *Soft constraint*: to spread conflicting exams across the timetables.

Table 2 presents the characteristics of the 11 problems in the dataset. The "conflict density" gives the density of elements with value 1 in the conflict matrix, where element $C_{ij} = 1$ if events i and j conflict, $C_{ij} = 0$ otherwise. The penalty of the timetable generated is the sum of costs per student, where costs w_i, $i \in 0, 1, 2, 3, 4$, indicate the timeslots between two conflict exams. More details can be found at http://www.asap.cs.nott.ac.uk/resources/data.shtml.

INSERT TABLE 2 HERE.

The rest of the paper is organised as follows. The GHH framework is formally presented in Section 2. A number of high level heuristics are investigated and analysed within this framework in Section 3. In Section 4, we present a hybrid GHH and analyse the search in two search spaces. Experimental results for both the course and exam benchmark timetabling problems are reported in Section 5. We present our conclusions on this work in Section 6.

2 The Graph Based Hyper-heuristic Framework

2.1 Graph Colouring Heuristics

Graph colouring heuristics [16] are constructive heuristics which order the events (courses or exams in timetabling) in terms of *difficulty* measures. These ordered events are then assigned, one by one from the most difficult ones, to construct the solutions (timetables). The basic assumption is that the most difficult events need to be scheduled earlier to avoid causing problems at a later stage. A list of the graph colouring heuristics widely employed in timetabling is presented in Table 3. For example, if Saturation Degree is used in an exam timetabling problem, the exams are ordered by the number of remaining valid timeslots in the

partial timetable during the solution construction. These ordered exams are scheduled one by one to the timetable being constructed.

INSERT TABLE 3 HERE.

As simple constructive heuristics, straightforward implementations of graph colouring heuristics on their own are not appropriate methodologies for addressing complex timetabling problems. Indeed, for some of the problem instances we are testing in this work, they failed to even generate feasible solutions on their own. However, recent research has shown that they can be very effectively employed as initialisation methods for meta-heuristics (e.g. [16]), or intelligently hybridised to solve complex timetabling and optimisation problems [4, 11, 21] for exam timetabling. This motivates us to investigate using hyper-heuristics to choose graph colouring heuristics for constructing high quality timetables.

2.2 The Graph Based Hyper-heuristic Approach

In our previous Graph Based Hyper-heuristic (GHH) [17], sequences of graph colouring heuristics (at the low level) were searched by a Tabu Search and applied to construct timetables. In most of the work on classical heuristics (such as graph colouring heuristics), solutions are built by assigning events in a specified order using a single or static strategy. These approaches often failed to obtain even feasible solutions. The GHH approach searches upon sequences of graph colouring heuristics, which order the events using different strategies during the solution construction. This can be seen as adaptively employing different and appropriate heuristics in different problem solving situations.

Figure 1 presents the process of the solution construction by using a sequence of graph colouring heuristics. At iteration i, the ith low level graph colouring heuristic in the sequence is employed to order the events (courses or exams) not yet scheduled by using the corresponding difficulty measuring strategy in Table 3. For example, if Saturation Degree in the heuristic sequence is employed at the current step of solution construction, the events that are not yet scheduled are ordered by the number of feasible timeslots available for the events at that time. Then the most difficult event (the 1st event in the ordered list) is scheduled into the timetable. In the next step, the next graph colouring heuristic in the heuristic sequence will be used to reorder the events that are left by using the corresponding (maybe different) difficulty measuring strategy. This process is repeated until a complete solution is constructed or no further feasible construction can be made. In this work the same solution construction process is used within the GHH framework.

INSERT FIGURE 1 HERE.

The role of the Tabu Search is to find, at a higher level, the best heuristic sequences to construct the best solutions. The penalty of the timetable constructed by the corresponding heuristic sequence is returned as the

objective value for the next step of the Tabu Search. The process is illustrated by the pseudo-code in Figure 2.

INSERT FIGURE 2 HERE.

The GHH approach [17] highlights two search spaces: the search space of high level heuristic sequences and the search space of solutions. Each of the heuristic sequences in the heuristic search space is used to construct a complete solution that corresponds to a point in the solution space. The quality of a solution in the solution space is used in the objective function for the high level heuristic in the heuristic search space. The GHH approach search is carried out upon combinations of difficulty measuring strategies (heuristic sequences) rather than actual solutions. The solutions constructed by similar ("close to each other's neighbourhood") heuristic sequences in the heuristic search space may be very different from each other in the solution space. This means that the GHH may cover solutions across different areas in the solution space by making neighbourhood moves in the high level search. Note that in most local search based timetabling methods, search is undertaken by moving just a few events in actual solutions (i.e. being just a few events *apart* within a local area). By making a neighbourhood move on heuristic sequences in the GHH high level heuristic, the solutions constructed have the potential to be far more *different* from each other. The search by the high level heuristic can be seen as jumping within the solution space. This is because the way of constructing the solutions is fundamentally changed. With the same amount of moves in the high level heuristic search, the mapping solutions are (potentially) widely distributed within the solution space.

2.2 Defining the GHH Framework

In this work, we extend the above approach by firstly presenting a formal definition of the GHH framework as follows:

Given an optimisation problem P, let H be the search space of P, and let f be the objective function that we are aiming to optimise. A moving operator O changes the incumbent solution $h \in H$ to its neighbourhood $N(h) \subset H$ by a moving strategy M. In terms of our timetabling problems, we consider the following problem and terminology.

- *Optimisation problem P*: to optimise the sequences of graph colouring heuristics h, which are employed to construct timetables concerning the objective function f. That is $P = \min f(h)$, $h \in H$.

- *Search space H*: consists of heuristic sequences $h \in H$.

- *Objective function f*: maps the heuristic sequences to the penalties of the corresponding timetables constructed (by evaluating the violations of soft constraints).

- *Moving operator O*: randomly changes any two heuristics in the heuristic sequence h.

- *Neighbourhoods N*: $N(h) \subset H$ are all the heuristic sequences *h* that can be reached by a moving operator *O* applied on *h*.

- *Moving strategy M*: a move is taken by using moving operator *O* and evaluating neighbourhoods *N*. This is defined by different high level heuristics. *Walks*, which accept moves to heuristic sequences of the same quality, are allowed in the GHH framework.

- *Low level heuristics*: elements (graph colouring heuristics) in the heuristic sequences *h*, which are used to construct timetables.

- *High level heuristics*: different search algorithms upon the search space *H* using different moving strategies *M*.

3 High Level Heuristics within the GHH Framework

We investigate a number of high level heuristics without any fine-tuning on their parameters (standard versions of the Steepest Descent method, Tabu Search, Iterated Local Search and Variable Neighbourhood Search) based on the unified framework defined above. The objective is to compare their performance under the unified framework with the aim of studying the heuristic search space, rather than fine-tuning them to get the best solutions. In practice, any of these algorithms can be employed to carry out local search. Five graph colouring heuristics (presented in Table 3) are employed as the low level heuristics in the framework. Of course, more constructive heuristics may be employed in the framework and we have investigated the effect of employing a different number of low level heuristics in the Tabu Search based GHH in [17].

In the context of testing different high level heuristics in this framework, there is no greedy local search on each complete solution that is generated at each move (as was the case in [17]). The aim is two fold. Firstly, the high level search in the GHH framework will not deal with the domain knowledge of the actual problems but rather it will work upon the objective values returned from the low level heuristics. All the necessary information about problems is dealt with by the low level heuristics. This establishes a certain level of generality in the GHH framework. Secondly, we would like to compare their behaviour within the unified framework in order to draw general conclusions about the GHH. Our aim is not to obtain the best possible results by putting effort into developing problem specific techniques. The goal is to develop methods which can be immediately used on a broad range of problems than can be achieved at the moment.

We will investigate the performance of different high level heuristics starting from the same initial points (initial heuristic sequences consisting of only Saturation Degree) and with the same computational cost (the same total number of evaluations). *Walks* (which accept heuristic sequences of the same quality) are always accepted because solutions of the same quality may be constructed by different heuristic sequences. Allowing

such moves within the search space of heuristic sequences gives more flexibility to the high level search. The methods that we employ are outlined in the following subsections.

3.1 Steepest Descent method (SDM)

We test a simple SDM where the best heuristic sequence among neighbourhoods is always selected if it is better than or the same as the current one. The search stops when there is no same or better neighbourhood, or after a set number of evaluations (which is the same as that used for the other high level heuristics) is carried out.

3.2 Iterated Local Search (ILS)

In the ILS [38], the search is re-started after a certain number of iterations of SDM. The re-starting point is made different from the previous one by changing the heuristics at the beginning of the heuristic sequence. Note that the earlier a different constructive heuristic is employed in the heuristic sequences, the larger the differences in the solutions generated by them. The aim is to explore wider regions of solution space. Again the same number of evaluations is set as the stopping condition.

3.3 Tabu Search (TS)

The TS we employ here is similar to that of our previous work [17]. The difference is that we do not carry out the greedy local search on the complete solutions at each step. The length of the tabu list is set to be 9, which is determined by empirical experience and suggested by classical settings in the literature [46]. The search stops after the set number of evaluations is carried out.

3.4 Variable Neighbourhood Search (VNS)

The motivation for employing VNS within the framework is to investigate the effect of employing more neighbourhood structures upon the performance of the high level search. The VNS employed is an iterative process where search is re-started after a certain number of walks are made by a standard VNS (see [34, 35]). The total number of evaluations is set as the same as it is for other high level heuristics.

We define a set of simple neighbourhood structures as randomly changing 2, 3, 4 or 5 heuristics in the sequence to any other graph colouring heuristics. These neighbourhood structures are simple, general and easy to implement and are widely employed in different VNS implementations. We do not employ a *swapping* neighbourhood because the search is upon heuristics rather than actual solutions. The two heuristics in the sequence do not have any link/relationship in the way that two events in actual solutions may have. Swapping two heuristics in the sequence has the same effect as that of changing the two corresponding heuristics in the sequence, which is included in the neighbourhoods we defined.

4 Hybridisations within the GHH Framework

Based on the GHH framework, we further explore how local search can be hybridised in terms of the search within the two search spaces. We also further investigate the hypothesis in [17] that only a subset of the solutions in the solution space can be sampled by the high level search.

4.1 Local Search within the GHH

We study two ways of hybridising the local search, which is a simple greedy search, within the GHH. They are marked in Figure 1 as (1) and (2), and are described below:

(1) *Local Improvement upon Complete Solutions (GHH1)*: At each step of the high level heuristic, a complete solution will be generated by a heuristic sequence. A greedy search is implemented on these complete solutions in the solution space to improve their quality in terms of their respective local optima.

(2) *Local Improvement during the Solution Construction (GHH2)*: During the solution construction by a heuristic sequence, at each step, the 1st event in the ordered list is scheduled into the partial solution. The greedy search is then implemented upon this partial solution to further reduce the costs to the lowest possible. In the next step of solution construction, the 1st event measured by the next graph colouring heuristic is scheduled based on the improved partial solution from the last step.

INSERT FIGURE 3 HERE.

We present, in Figure 3, the pseudo-code of the hybrid GHH within a general framework without specifying a particular high level search algorithm. Instructions marked with a * are optional depending on the particular high level heuristics or the hybrid local search employed.

4.2 Two Search Spaces within the Hybrid GHH

In the hybrid GHH, two search spaces are involved. They are the search space of high level heuristics upon heuristic sequences, and the search space of the local search upon actual solutions. That is, not only the sequences in the search space of heuristics are searched to generate the best possible solutions mapping to certain solutions in the solution space, but also the mapping solutions are improved by the local search in the search space of actual solutions. The latter is similar to most of the local search algorithms in the literature which operate upon a search space of solutions.

Table 4 presents the characteristics of these two fundamentally different search spaces in the GHH framework. Not only the representations, but also the upper bounds of the size of the search spaces (including infeasible points) are different. The objective functions are different. They evaluate the quality of the solutions generated by the corresponding sequences, or evaluate the delta costs incurred in the solutions by

the neighbourhood operators, respectively. The neighbourhood operators are also different depending on the high level heuristics and local search employed in the hybrid GHH framework.

INSERT TABLE 4 HERE.

Also note that in the GHH, at each step, five possible graph colouring heuristics may be used to assign each event. The upper bound of the size of the heuristic search space is then 5^e. This is much smaller than that of the solution space (which is t^e, where t is the number of timeslots) because, in practice, the number of timeslots is usually much higher than 5. This implies that the search of the high level heuristics is carried out within a much smaller search space. This also further shows that not all of the solutions in the solution space correspond to sequences in the heuristic space.

4.3 Exploring and Exploiting within the Hybrid GHH

The aim of hybridising local search methods within the GHH framework is twofold. Firstly, along with the exploration by the high level heuristics (*jumping* within different areas of the solution space), the local search moves (in the solution space) to local optima. Secondly, as not all of the solutions can be mapped to the heuristic sequences, performing a local search can make more solutions in the solution space reachable.

At a higher level, the overall idea of the exploration and exploitation by the hybridisation in GHH is similar to that of a Memetic Algorithm or genetic local search [36], where recombination and/or mutation on population facilitates the exploration of the search, while the local search exploits within local regions. The difference is that in our hybrid GHH the high level heuristic explores the search space of solutions by searching heuristic sequences in the manner of local search at a higher level. Also in Memetic Algorithms, an initial population is needed, while the GHH constructs solutions iteratively, thus is not sensitive to initialisation methods. This makes the hybrid GHH much simpler yet still capable of exploring and exploiting the search space simultaneously.

5 Experimental Results on Benchmark Timetabling Problems

We carried out two sets of experiments to test the GHH which a) employs different high level heuristics (see Section 3) and b) is hybridised with local search (see Section 4). Two sets of the widely studied benchmark course and exam timetabling problems are used (see Section 1.3). These datasets are very different and have been widely employed by a number of state-of-the-art approaches (but have not been tackled together by one approach). Our aim with these experiments is not to *beat* the state-of-the-art approaches in the literature (although the results are competitive with the best results reported), but to present the potential of this more generic methodology to be easily employed and to perform adaptively on a range of different timetabling or

optimisation problems. Fundamental issues such as neighbourhood structures and search spaces are also analysed to gain a deeper understanding of the technique.

5.1 Analysis on the High Level Search Space of Heuristics

Different high level heuristics described in Section 3 are tested on both of the benchmark problems within the unified GHH framework. Due to the fact that there is no domain knowledge required in the framework, the only difference in the GHH upon these two different problems is the evaluation functions that are used to evaluate the solutions generated. This can be easily switched between different problems and this is the key achievement of the work: to develop a methodology that can be easily switched between different problems. In both sets of experiments, the best and average results obtained from five runs with distinct seeds are reported. As the same number of evaluations is set as the stopping condition for all the high level heuristics, similar computational time is observed for each of these approaches. It is important to note that the purpose of this section is to investigate the employment of different methods on the heuristic space. We use the results of this section to produce the better results which are generated by the hybridised method in the next section.

5.2.1 Course Timetabling Problems

Table 5 presents the best and average results by the GHH employing the Steepest Descent Method (SDM), Iterated Local Search (ILS), Tabu Search (TS) and Variable Neighbourhood Search (VNS) as the high level heuristics for the benchmark course timetabling problems. It can be seen that within the unified GHH framework, most of the high level heuristics (except SDM) perform similarly. VNS obtained the best results (on a small scale) on 6 out of 11 problems, and the best average results on 5 out of 11 problems.

Another observation is that when GHH is applied to course timetabling problems, some of the soft constraints (such as soft constraint 3 in Table 1) cannot be evaluated accurately during the solution construction until a complete solution is generated. This may affect GHH's ability to obtain high quality solutions for some problems.

INSERT TABLE 5 HERE.

5.2.2 Exam Timetabling Problems

The results on the benchmark exam timetabling problems by the GHH employing SDM, ILS, TS and VNS are presented in Table 6. It can be observed that, again, all high level heuristics (except SDM) perform quite similarly. ILS obtained the best results on 6 out of 11 problems and the best average results on 5 out of 11 problems. VNS is the 2nd best algorithm (again on a small scale though) on these problems, obtaining the best results for 4 out of 11 problems and the best average results for 5 out of 11 the problems.

INSERT TABLE 6 HERE.

5.2.3 Discussion

The comparisons between the different high level heuristics on both course and exam timetabling problems have shown that high level heuristic moving strategies are not crucial within the unified GHH framework. Elaborate algorithms such as TS are not effective for searching in the search space of heuristics. Iterative techniques such as ILS and VNS (which is, overall, an iterative process - see Section 3.4) were shown to be effective on both the course and exam timetabling problems tested.

The reason is that very large and different areas of the solution space can be covered by relatively small moves in the heuristic space. With the search in the heuristic space, the chance of the high level search having better or same quality *neighbourhoods* is higher than that of local search methods. During the high level heuristic search, it was observed that a large number of walks were performed before the search obtained the next better heuristic sequences. The search space of the high level heuristic is more likely to be smooth and to contain large areas of plateaus where different heuristic sequences can produce similar quality solutions.

Due to the characteristics of the high level search space of heuristic sequences, we thus conclude that the role of high level heuristics within the GHH is to *search quickly within limited areas and to restart iteratively at different areas*. More parts of the solution space (even disconnected from each other) can thus be searched, which increases the chance of obtaining more high quality solutions in a limited time.

5.3 Analysis of the Hybridisations within the GHH Framework

In this section, the two hybridisations of the GHH (see GHH1 and GHH2 in Section 4.1) are tested on both benchmark course and exam timetabling problems. Due to the above observations (see subsection 5.2.3), we employ ILS as the high level heuristic.

5.3.1 Course Timetabling Problems

Table 7 presents the results of our hybrid GHH, together with the best results reported in the literature by different approaches. We can observe that both of the hybrid GHH methods work well. In particular, GHH2 obtained the best and best average results on all of the small instances and two of the medium instances among all of the approaches reported in the literature. During the experiments of GHH2, we observed that high quality solutions can usually be obtained fairly quickly after each re-start of the high level ILS. This, in practice, can significantly reduce the computational time of this hybrid GHH. Also note that for GHH2, the best and average results on all the problems are very close, meaning that the approach is consistent over different runs.

INSERT TABLE 7 HERE.

As explained above, because GHH searches for heuristic sequences to construct solutions, some of the soft constraints (such as soft constraint 3 in Table 1) cannot be accurately evaluated on the partial solution built at that time. By hybridising local search upon actual solutions, these soft constraints can be effected on a complete or a partial solution during the solution construction.

5.3.2 Exam Timetabling Problems

Table 8 presents the results on the benchmark exam timetabling problems by the two hybrid GHH approaches. Again, GHH2 outperforms GHH1 over all of the problems on average results, and 9 out of 11 problems on the best results. Hybridising local greedy search to carry out exploitive search upon partial solutions during the solution construction is observed to be more effective than that of on a complete solution. Again the GHH2 is very consistent on all of the runs with distinct seeds.

INSERT TABLE 8 HERE.

It can be observed that the best results reported in the literature were obtained by different approaches over the years. Among the 8 approaches compared (which have obtained the best results in the literature on the benchmarks), our hybrid GHH obtained competitive results. However, the most important point to make here is that all of the other approaches were specifically designed for the exam timetabling problem. Only our method, which does not require parameter tuning, can also work well on the course timetabling problem by simply changing its evaluation function.

6 Conclusions and Future Work

This paper defined a unified graph based hyper-heuristic (GHH) framework. It investigated different high level heuristics on both benchmark course and exam timetabling problems. Hybridisations of the GHH with local search methods were also investigated.

We studied the search space of high level heuristics within the GHH framework. Experimental results indicate that iterative techniques such as Iterated Local Search and Variable Neighbourhood Search are more effective than Tabu Search and Steepest Descent Method within this GHH framework on both course and exam timetabling problems. It is observed that by employing neighbourhood moves within the heuristic search space, the high level search is capable of *jumping* within the solution space of problems. The heuristic sequences cover only a subset (but well distributed areas) of the solution space. It is also observed that the search space of heuristics is relatively smooth (i.e. it contains a large number of plateaus where solutions with the same quality exist). We thus suggest that the role of the high level heuristic in GHH is to search

within the limited areas quickly and to explore as widely as possible the solution space by re-starting from different heuristic sequences within a limited computational time.

We investigated two ways of hybridising the GHH with local search. Of particular interest is the hybridisation of local search during the solution construction, which can quickly obtain good results for both benchmark course and exam timetabling problems compared with the different approaches that appeared in the literature over the years. Compared with the basic GHH, the hybridisation of local search during the solution construction presented significant improvement on solution quality and computational time.

The comparisons of the characteristics of the two search spaces (of heuristics and of solutions) further strengthen the views that not all of the solutions in the solution space can be mapped to the heuristic sequences in the heuristic search space. This implies that the high level search is carried out within a much smaller search space (of heuristics) but is still able to cover wide areas of the solution space. The hybrid GHH has the ability of exploring and exploiting within the solution space simultaneously by searching within the two search spaces.

The hybrid GHH is a more general framework than other timetabling methods. Simple and fast search techniques can be employed as the high level heuristic and the local search on the solution space without much tuning effort. In fact, the basic methodology can be applied to any problems that can be modelled as graph colouring problems. Testing on a wider range of application domains such as general timetabling and scheduling problems represents a significant direction in our future research. The hyper-heuristic search has a simple structure and is focused on high level heuristics, leaving much scope for hybridisation with other techniques. Further in-depth performance analysis and understanding may also lead to general observations benefiting general meta-heuristic research. For example, knowledge of the good sequences that generate high quality solutions can be extracted and collected in helping build fundamentally more general approaches for timetabling and optimisation problems. As an iterative constructive approach, the computational time of GHH is usually much larger than that of local search improvement algorithms. More knowledge and better understanding of the GHH framework may further improve the efficiency of this general approach for a range of optimisation problems.

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. Abdullah S., Burke E.K. and McCollum B.: An Investigation of a Variable Neighborhood Search Approach for Course Timetabling. Proceedings of the 2nd Multidisciplinary Conference on Scheduling: Theory and Applications, 18-21 July, NY, USA, 413-427, July, 2005.

3. Abdullah S., Burke E.K. and McCollum B.: Using a Randomised Iterative Improvement Algorithm with Composite Neighborhood Structures for University Course Timetabling. In: Doerner K.F., Gendreau M., Greistorfer P., Gutjahr W.J., Hartl R.F. and Reimann M. (eds.). Computer Science Interfaces Book Series, Springer Operations Research, 2006.

4. Asmuni H., Burke E.K. and Garibaldi J.: Fuzzy Multiple Ordering Criteria for Examination Timetabling. In: Burke E.K. and Trick M. (eds.): Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3616, 334-353, 2005.

5. 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: Selected Papers from the 1st International Conference. Springer Lecture Notes in Computer Science 1153. 22-45. 1996.

6. Bilgin B., Özcan E. and Korkmaz E.E.: An Experimental Study on Hyper-Heuristics and Exam Scheduling, In: Burke E.K. and Rudová H. (eds.) Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3867, 394-412.

7. Brailsford S.C., Potts C.N. and Smith B.M..: Constraint satisfaction problems: Algorithms and applications. *European Journal of Operations Research*, **119**: 557- 581, 1999.

8. Brucker P. and Knust S. Complex Scheduling. Springer, 2006.

9. 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.

10. Burke E.K., De Causmaecker P., Vanden Berghe G. and Van Landeghem H.: The State of the Art of Nurse Rostering. *Journal of Scheduling*, **7**(6): 441-499, 2004.

11. 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.

12. 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. (eds). *Handbook of Meta-Heuristics*, Kluwer 457-474, 2003.

13. Burke E.K., Jackson K., Kingston J.H. and Weare R.: Automated University Timetabling: The State of the Art. *The Computer Journal*, **40**(9): 565-571, 1997.

14. Burke E.K. and Kendall G. (eds.): Introductory Tutorials in Optimisation, Decision Support and Search Methodology, Springer, 2005.

15. Burke E.K., Kendall G. and Soubeiga E.: A Tabu Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*. **9**(6): 451-470, 2003.

16. 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.

17. Burke E.K., McCollum 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.

18. Burke E.K. and Newall J.: A Multi-Stage Evolutionary Algorithm for the Timetabling Problem. *IEEE Transactions on Evolutionary Computation*. **3**(1): 63-74, 1999.

19. Burke, E.K. and Newall, J.: Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings. *Annals of operations Research*, **129**: 107-134, 2004.

20. Burke E.K. and Petrovic S. Recent Research Directions in Automated Timetabling. *European Journal of Operational Research*, **140**(2): 266-280, 2002.

21. Burke E.K., Petrovic S. and Qu R.: Case Based Heuristic Selection for Examination Timetabling. *Journal of Scheduling*, **9**(2): 99-113, 2006.

22. Caramia M., DellOlmo P. and Italiano G.: New Algorithms for Examination Timetabling. In: Naher, S., Wagner, D. (eds.) Algorithm Engineering, Lecture Notes in Computer Science 1982, 230-241, 2001.

23. Carter M. and Laporte G.: Recent Developments in Practical Exam Timetabling. In: Burke E.K. and Ross P. (eds.): Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1153, 3-21, 1996.

24. Carter M. and Laporte G.: Recent Developments in Practical Course Timetabling. In: Burke E.K. and Ross P. (eds.): Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1153, 3-19, 1998.

25. Carter M., Laporte G. and Lee S.: Examination Timetabling: Algorithmic Strategies and Applications. *Journal of Operational Research Society*, **47**: 373-383, 1996.

26. Casey S. and Thompson J.: GRASPing the Examination Scheduling Problem. In: Burke, E. and Causmaecker, P. (eds.): Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2740, 232-246, 2003.

27. Cicirello V.A. and Smith S.F.: Heuristic Selection for Stochastic Search Optimization: Modeling Solution Quality by Extreme Value Theory. In: Principles and Practice of Constraint Programming CP 2004, 197-211, 2004.

28. Dowsland K., Soubeiga E. and Burke E.K.: A Simulated Annealing Hyperheuristic for Determining Shipper Sizes. *European Journal of Operational Research*, **179**(3): 759-774, 2007.

29. Easton K., Nemhauser G. and Trick M.: Sports Scheduling. In: Leung J. (ed.). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapter 52, CRC Press, 2004.

30. Eley M.: Ant Algorithms for the Exam Timetabling Problem. In: Burke E.K. and Rudova H. (eds). Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3867, 364-382, 2007.

31. Di Gaspero L. and Schaerf A.: Tabu Search Techniques for Examination Timetabling. In: Burke E.K. and Erben W. (eds.): Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2079, 104-117, 2001.

32. Gaw A., Rattadilok P. and Kwan R.S.: Distributed Choice Function Hyperheuristics for Timetabling and Scheduling. In: Burke E.K. and Trick M. (eds.): Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3616, 51-67, 2005.

33. Gomes C.P. and Selman B.: Algorithm Portfolios. *Artificial Intelligence*, **126**(1-2): 43 - 62, 2001.

34. Hansen P. and Mladenovic N.: Variable Neighbourhood Search: Principles and Applications. European *Journal of Operational Research*, **130**: 449-467, 2001.

35. Hansen P. and Mladenovic N.: Variable Neighbourhood Search. In: Burke E.K. and Kendall G. (eds.) *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*. Springer, Chapter 8, 211-238, 2005.

36. Krasnogor N., Aragon A. and Pacheco J.: Memetic Algorithms. In: Marti R. and Alba E. (eds.), Metaheuristics in Neural Networks Learning. Kluwer, 2006.

37. Kwan R.: Bus and Train Driver Scheduling. In: Leung J. (ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapter 51, CRC Press, 2004.

38. Lourenco H.R., Martin O. and Stutzle T.: Iterated Local Search. In: Glover F. and Kochenberger G. (eds.), *Handbook of Metaheuristics*. 321-353. 2003.

39. Merlot L., Boland N. Hughes B. and Stuckey P.: A Hybrid Algorithm for the Examination Timetabling Problem. In: Burke, E. and Causmaecker, P. (eds.): Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2740, 207-231, 2002.

40. Meyers C. and Orlin J.B.: Very Large-Scale Neighborhood Search Techniques. In: Burke E.K. and Rudova H. (eds.) Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3867, 24-39, 2007.

41. McCollum B.G.C.: A Perspective on Bridging the Gap in University Timetabling. In: Burke E.K. and Rudová H. (eds.), Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 3867, 3-23.

42. Nareyek A.: Choosing Search Heuristics by Non-Stationary Reinforcement Learning. In: Resende M.G.C. and de Sousa J.P. (eds.), Metaheuristics: Computer Decision-Making, Kluwer Academic Publishers, 523-544. 2003.

43. Nonobe K. and Ibaraki T.: A Tabu Search Approach to the Constraint Satisfaction Problem as a General Problem Solver. *European Journal of Operational Research*, **106**: 599-623, 1998.

44. Petrovic S. and Burke E.K.: University Timetabling. In: Leung J. (ed.) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapter 45, 2004.

45. Qu R., Burke E.K., B. McCollum, L.T.G. Merlot, and S.Y. Lee. A Survey of Search Methodologies and Automated System Development for Examination Timetabling. Accepted by *Journal of Scheduling*, 2008. DOI: 10.1007/s10951-008-0060-1.

46. Reeves C.R. (ed.): *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Scientific Publications. 1993.

47. Ross P.: Hyper-heuristics. In: Burke E.K. and Kendall G. (eds.) *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*. Springer, Chapter 17, 529-556, 2005.

48. Ross P., Marin-Blazquez J. and Hart E.: Hyper-heuristics Applied to Class and Exam Timetabling Problems. Proceedings of the 2004 congress on evolutionary computation (CEC 2004), 1691-1698.

49. Schaerf A.: A Survey of Automated Timetabling. *Artificial Intelligence Review*. **13**(2): 87-127, 1999.

50. Socha K., Knowles J. and Sampels M.: A Max-Min Ant System for the University Course Timetabling Problem. In: Proceedings of the 3rd International Workshop on Ant Algorithms, Lecture Notes in Computer Science 2463, 1-13, 2002.

51. Terashima-Marin H., Ross P., Valenzuela-Rendon M.: Evolution of Constraint Satisfaction Strategies in Examination Timetabling. Proceedings of the Genetic and Evolutionary Computation Conference, 635-642, 1999.

52. Thompson J. and Dowsland K.: A Robust Simulated Annealing Based Examination Timetabling System. *Computers & Operations Research*, **25**: 637-648, 1998.

53. de Werra D.: An Introduction to Timetabling. *European Journal of Operational Research*, **19**: 151-162, 1985.

54. White G. M.: Constrained Satisfaction, Not So Constrained Satisfaction and the Timetabling Problem. In: Burke E.K. and Erben W. (eds.) Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, keynote, 32-54, 2000.

55. White G.M., Xie B.S. and Zonjic S.: Using Tabu Search with Longer Term Memory and Relaxation to Create Examination Timetables. *European Journal of Operational Research*, **153**(16): 80-91, 2004.

Table 1 Constraints of the benchmark course timetabling problems

Hard constraints	Soft constraints
no students can be scheduled to more than one event at the same time	a student has a class in the last timeslot of the day
the room meets all features required by the event	a student has more than two classes in a row
no more than one event is allowed per room and per timeslot	a student has only one class on a day
room capacity is respected	

Table 2 Characteristics of the benchmark exam timetabling problems

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta93 I	yok83 I
exams	682	543	190	81	461	381	139	261	184	622	181
timeslots	35	32	24	18	20	18	13	23	10	35	21
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

Table 3 Difficulty measuring strategies based upon graph heuristics for timetabling problems

Graph Heuristics	Difficulty Measuring Strategy that Orders Events
Largest Degree	decreasingly by the number of conflicts the event has with the other events
Largest Weighted Degree	the same as above but weighted by the number of students involved in the events
Largest Enrolment	decreasingly by the number of enrolments in the events
Colour Degree	decreasingly by the number of conflicts the event has with those already scheduled
Saturation Degree	increasingly by the number of feasible timeslots available for the event at that time

Table 4 Characteristics of the search spaces of the heuristics and solutions in the hybrid GHH

Search Space	Heuristics	Solutions
Representation	sequences of graph colouring heuristics	actual timetables
Size (Upper Bound)	h^l (l: length of the sequence, h: number of graph colouring heuristics)	t^e (t: number of timeslots, e: number of events)
Neighborhood Operator	randomly change two heuristics in the sequence	move events in the timetable to other timeslots
Objective Function	penalty of timetables constructed by heuristic sequence	penalty of timetables, or difference of costs caused by moving events in the timetable

Table 5 Different high level heuristics in the GHH framework on benchmark course timetabling problems (s1-s5: small problems; m1-m5: medium problems). The best results and the best average results are highlighted and italicised for each problem, respectively. "x%": the percentage of runs where no feasible solutions can be obtained.

	s1	s2	s3	s4	s5	m1	m2	m3	m4	m5	large
SDM best	7	**8**	**3**	**6**	10	368	100%	367	**356**	195	100%
SDM avg	10.8	15.6	*5*	11.8	12.2	382.5	100%	383	*374.5*	194.5	100%
SDM time (s)	15	38	10	8	30	3823	3672	3752	3637	1989	4013
ILS best	**6**	9	4	**6**	8	373	461	375	374	172	**1132**
ILS avg	*8.8*	*13.2*	5.4	*7.6*	12	375	480.5	377.5	380.5	179.7	*1144* 60%
ILS time (s)	32	47	15	11	23	3656	3018	3382	3451	1822	3811
TS best	11	11	5	11	16	496	533	460	529	214	1164
TS avg	12.2	16.4	9.2	12.2	18.2	511.5	533 80%	468	539	236	1164 80%
TS time (s)	12	18	9	7	19	3326	2996	3160	3280	1650	3564
VNS best	7	12	4	**6**	**6**	**346**	**433**	**359**	370	**156**	1148
VNS avg	10	14.8	5.2	8	*10.6*	*365*	*443* 40%	*369.5*	377.5	*165.5*	1148 80%
VNS time (s)	32	45	16	10	30	3920	3723	3856	3667	2013	4079

Table 6 Different high level heuristics in the GHH framework on benchmark exam timetabling problems (the best and best average results are highlighted and italicised, respectively.)

	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta93 I	yor83 I
SDM best	5.44	4.87	**35.54**	12.59	15.25	13.01	160.3	9.01	31.77	3.61	42.77
SDM avg	6.18	5.3	*36.8*	12.74	15.63	13.51	163.7	9.37	32.6	4.5	43.6
SDM time (s)	15367	8001	584	22	2502	1722	69	1597	87	8018	426
ILS best	**5.3**	4.77	38.39	12.72	**15.09**	12.72	**159.2**	8.74	**30.32**	**3.32**	**40.24**
ILS avg	*6.01*	5.18	39.58	13.01	15.35	13.1	*161.6*	8.92	*31.3*	*4.01*	*43.15*
ILS time (s)	17334	8200	617	31	2629	1832	73	1638	100	10464	527
TS best	5.43	4.94	38.19	12.36	15.97	13.25	165.7	8.87	32.12	3.52	41.3
TS avg	6.3	5.34	45.56	14.6	19.55	14.29	169.1	9.67	37.02	4.38	47.97
TS time (s)	20393	9111	649	32	2768	1970	80	1800	100	10464	527
VNS best	5.4	**4.7**	37.29	**12.23**	15.1	**12.71**	159.3	**8.67**	30.23	3.56	43.0
VNS avg	6.1	*5.1*	38.63	*12.72*	*15.24*	*13.06*	163.3	*8.88*	31.7	4.05	43.93
VNS time (s)	16321	8107	672	42	2531	1653	47	1721	677	9210	501

Table 7 State-of-the-art approaches and GHH hybridising local search (GHH1: on complete solutions; GHH2: during solution construction) on benchmark course timetabling problems. "x/y": the percentage x that cannot obtain feasible solutions, and the best result y of feasible solutions. The best results and the best average results are highlighted and italicised, respectively.

	s1	s2	s3	s4	s5	m1	m2	m3	m4	m5	large
GHH1 best	**0**	**0**	**0**	**0**	**0**	257	259	**192**	235	**112**	0.8/1132
GHH1 avg	*0.2*	*0.6*	*0*	*0.4*	*0.1*	*261*	*273*	*214.5*	*242*	*116*	*1135*
GHH1 time (s)	50	54	48	45	65	19411	15750	18512	18782	9725	20328
GHH2 best	2	2	1	1	**0**	310	419	332	324	162	0.8/1162
GHH2 avg	2.6	2.8	1	3	2.6	323	428	345	335	182	1162
GHH2 time (s)	155	218	240	171	260	62115	50403	57387	65821	36955	81148
ANT [50]	1	3	1	1	0	195	184	248	**164.5**	219.5	851.5
CNS [3]	**0**	**0**	**0**	**0**	**0**	242	**161**	265	181	151	**757**
TS-HH [15]	1	2	**0**	1	**0**	**146**	173	267	169	303	1166 80%
RRLS [50]	8	11	8	7	5	199	202.5	77.5%	177.5	100%	100%
VNS [2]	**0**	**0**	**0**	**0**	**0**	317	313	357	247	292	929

Table 8 State-of-the-art approaches and GHH hybridising local search (GHH1: on complete solutions; GHH2: during solution construction) on benchmark exam timetabling problems. A selection of appropriate methods which includes those that generate the best results among all of the approaches in the literature are highlighted.

|	car91 I	car92 I	ear83 I	hec92 I	kfu93	lse91	sta83 I	tre92	ute92	uta93 I	yor83 I
GHH2 best	5.16	4.16	35.86	11.94	14.79	11.15	159	8.6	28.3	3.59	41.81
GHH2 avg	5.21	4.20	36.2	12.1	15.01	11.24	160.81	8.65	28.64	3.62	41.96
GHH2 time (s)	26001	11666	740	105	3417	2015	128	2293	131	10045	641
GHH1 best	5.3	4.77	38.39	12.01	15.09	12.72	159.2	8.74	30.32	3.42	40.24
GHH1 avg	6.01	5.18	39.58	12.33	15.35	13.1	161.6	9.0	31.3	4.01	43.15
GHH1 time (s)	13684	6553	462	70	1887	1125	72	1433	101	5429	340
[1]	5.21	4.36	34.87	10.28	**13.46**	10.24	159.2	8.7	26	3.63	**36.2**
[4]	5.2	4.52	37.02	11.78	15.81	12.09	160.42	8.67	27.78	3.57	40.66
[19]	**4.6**	**4.0**	37.05	11.54	13.9	10.82	168.73	8.35	25.83	**3.2**	36.8
[9]	4.8	4.2	35.4	10.8	13.7	10.4	159.1	**8.3**	25.7	3.4	36.7
[22]	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	158.2	9.4	**24.4**	3.5	36.2
[25]	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	25.8	3.5	41.7
[31]	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	29.0	4.2	42.0
[39]	5.1	4.3	35.1	10.6	13.5	10.5	**157.3**	8.4	25.1	3.5	37.4
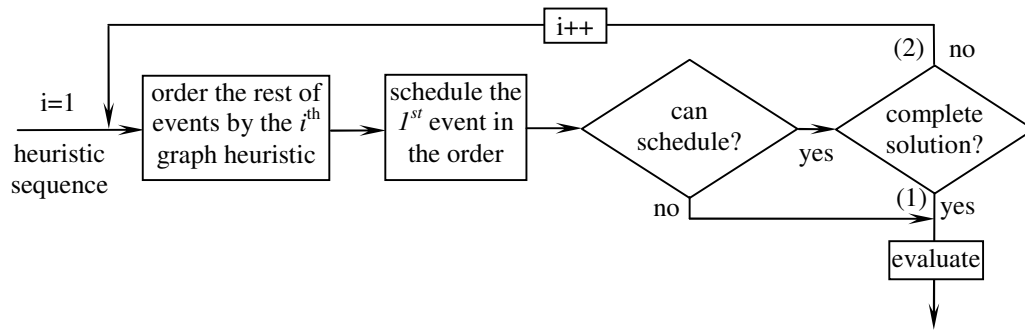
Figure 1 Solution construction by a sequence of graph heuristics

```
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 complete solution using heuristic sequence h (see Figure 1)
             if penalty of solution c < the least penalty cg obtained
               save the best solution, cg = c
               update the tabu list
               hl = h
     //end if
//end of Tabu Search
output the best solution with penalty of cg
```

Figure 2 Pseudo-code of the graph based hyper-heuristic

```
initialisation of the heuristic sequence hl = {h1 h2 … hk} //k: number of events

// high level search upon heuristic sequences
for i = 0 to i = the number of iterations
     *multiple starts in Iterated Local Search //optional
     h = change upon heuristic sequence hl //a move in the high level search

     //construct solution using heuristic sequence h
     for j = 0 to j = k //k: length of heuristic sequence
        schedule an event using heuristic hj
        *(2) local improvement on the partial solution (see (2) in Figure 1)

     *(1) local improvement on complete solution (see (1) in Figure 1)
     if penalty of solution c <= the least penalty cg obtained
        save the best solution according to acceptance criteria in the high level
        search, cg = c
        *update tabu list //optional for Tabu Search
        hl = h
//end of high level search
output the best solution with penalty of cg
```

Figure 3 The hybrid GHH framework

List of Figures

List of Tables