In press. **European Journal of Operational Research**, 2006

A Graph-Based Hyper-Heuristic for Educational Timetabling Problems

EDMUND K BURKE[1], BARRY MCCOLLUM[2], AMNON MEISELS[3], SANJA PETROVIC[1], RONG QU[1]*

[1]*Automated Scheduling Optimization & Planning Group, University of Nottingham, Nottingham, NG8 1BB, UK*

[2]*School of Computer Science, Queen's University Belfast, Belfast BT7 1NN, UK*

[3]*Department of Computer Science, Ben-Gurion University, Beer-Sheva 84 105, ISRAEL*

**corresponding author, email: rxq@cs.nott.ac.uk, telephone: ++44 115 9514215, fax: ++44 115 8467591*

Key words: heuristics, graph heuristics, hyper-heuristics, tabu search, timetabling

Abstract

This paper presents an investigation of a simple generic hyper-heuristic approach upon a set of widely used constructive heuristics (graph coloring heuristics) in timetabling. Within the hyper-heuristic framework, a Tabu Search approach is employed to search for permutations of graph heuristics which are used for constructing timetables in exam and course timetabling problems. This underpins a multi-stage hyper-heuristic where the Tabu Search employs permutations upon a different number of graph heuristics in two stages. We study this graph-based hyper-heuristic approach within the context of exploring fundamental issues concerning the search space of the hyper-heuristic (the heuristic space) and the solution space. Such issues have not been addressed in other hyper-heuristic research. These approaches are tested on both exam and course benchmark timetabling problems and are compared with the fine-tuned bespoke state-of-the-art approaches. The results are within the range of the best results reported in the literature. The approach described here represents a significantly more generally applicable approach than the current state of the art in the literature. Future work will extend this hyper-heuristic framework by employing methodologies which are applicable on a wider range of timetabling and scheduling problems.

1. Introduction

1.1 Timetabling Problems

Timetabling has attracted the attention of the Operational Research and Artificial Intelligence research communities for more than 40 years. The general timetabling problem comes in many different guises including nurse rostering (e.g. Cheang et al, 2003, Burke et al. 2004), sports timetabling (e.g. Easton, Nemhauser and Trick, 2004), transportation timetabling (e.g. Kwan, 2004) and university timetabling (Schaerf, 1999, Burke and Petrovic, 2002, Petrovic and Burke, 2004), etc. Further examples of papers dealing with these kinds of problems can be found in (Burke and Ross, 1996, Burke and Carter, 1998, Burke and Erben, 2000, Burke and De Causmaecker, 2002, Burke and Trick, 2004). Perhaps the most widely studied class of timetabling problem is that of educational timetabling. A wide variety of papers describing a broad spectrum of educational timetabling methodologies has appeared in the literature over the years. Overviews of the literature can be found in the following papers (Carter and Laporte, 1996&1998, Bardadym, 1996; Burke, Jackson et al, 1997, Schaerf, 1999, Burke and Petrovic, 2002, Petrovic and Burke, 2004).

In a general educational timetabling problem, a set of events (e.g. courses and exams, etc) need to be assigned into a certain number of timeslots (time periods) subject to a set of constraints, which often makes the problem very hard to solve in real-world circumstances. Constraints can usually be divided into two types:

- *Hard constraints* have to be satisfied under any circumstances. For example, in exam timetabling, two exams with common students involved cannot be scheduled into the same timeslot. Timetables with no violations of hard constraints are called *feasible* solutions.
- *Soft constraints* need to be satisfied as much as possible. For example, in exam timetabling, exams taken by common students often need to be spread out over the timeslots so that students do not have to sit in two exams that are too close to each other. Due to the complexity of the real-world timetabling problem, the soft constraints

may need to be relaxed since it is not usually possible to generate solutions without violating some of them. Soft constraints are usually within the cost evaluation function to evaluate how good the solutions are.

1.2 Approaches and Techniques in Timetabling Problems

The early days of research in educational timetabling investigated such approaches as graph heuristics (see de Werra, 1985, Burke, Kingston and de Werra, 2003) and integer linear programming (see Carter, 1986). Some of the early techniques are either impractical or too simple to solve complex or large timetabling problems. Constraint based techniques have been employed over the years to address timetabling problems (e.g. Deris et al, 1997, Banks, Beek and Meisels, 1998, Nonobe and Ibaraki, 1998). Recently, meta-heuristic search techniques (see Glover and Kochenberger, 2003) have been investigated and seem to have been very successful in solving a variety of timetabling problems. These include Tabu Search (e.g. Costa, 1994, Di Gaspero and Schaerf, 2000), Simulated Annealing (e.g. Dowsland, 1998, Abramson, Krishnamoorthy and Dang, 1999) and Evolutionary Algorithms (e.g. Burke, Newall and Weare, 1996&1998, Burke and Newall, 1999, Erben, 2000, Lewis and Paechter, 2004, Côté, Wong and Sabourin, 2005). Other new approaches and methodologies for timetabling problems have also been studied as more problem solving experience is collected and new technologies provide new breakthroughs. These include Case-Based Reasoning (Leake, 1996) on educational timetabling (Burke, MacCarthy et al. 2000, 2001, 2003&2005, Burke, Petrovic and Qu, 2006) and on nurse rostering (Beddoe and Petrovic, 2005), fuzzy methodology on exam timetabling (Asmuni, Burke and Garibaldi, 2004), and hyper-heuristics on timetabling (Burke, Kendall and Soubeiga, 2003, Gaw, Rattadilok and Kwan, 2004, Burke, Dror et al, 2005, Burke, Petrovic and Qu, 2006, Qu and Burke, 2005).

The present work concerns educational timetabling including both exam and course timetabling problems (Carter and Laporte, 1996, Carter and Laporte, 1998). The state-of-the-art approaches in educational timetabling usually employ specially tailored heuristic/meta-heuristic approaches (e.g. Carter, Laporte and Lee, 1996, Di Gaspero and Schaerf, 2000, Caramia, Dell'Olmo, and Italiano, 2001, Burke and Newall, 2002, Merlot et al, 2002, Socha, Knowles and Sampels, 2002, Asmuni, Burke and Garibaldi, 2004, Abdullah et al, 2004, Burke, Bykov et al, 2004). These approaches invest a significant amount of development effort in the production of "fine tuned" techniques that are specially built for the particular problems in hand. The objective of the present paper is to develop an approach which is more widely applicable and fundamentally more general than the approaches mentioned above. Our goal is not to "beat" them but to obtain comparable results by only employing general and simple techniques which can be applied to a wider range of scheduling problems.

1.3 Hyper-heuristics on Scheduling and Timetabling Problems

The development of hyper-heuristics is motivated by the goal of aiming at an increased level of generality for automatically solving a range of problems (see Burke, Kendall et al. 2003). A hyper-heuristic can be seen as an algorithm (on a *higher level*) which "picks" appropriate heuristics (at a *lower level*) to be applied to the problems in hand. A hyper-heuristic is concerned with the exploration of a search space of heuristics instead of dealing directly with solutions to the problem. In hyper-heuristic research on timetabling and scheduling, different techniques have been investigated as the low level and high level search strategies in solving the problems. We can categorize the work in hyper-heuristics (in terms of the "low level" heuristics employed) into two groups: improvement techniques and constructive techniques.

1.3.1 Improvement Techniques within Hyper-heuristics

In a hyper-heuristic, *move* strategies are usually employed as the low level heuristics to search for solutions to timetabling and scheduling problems. Kendall, Cowling and Soubeiga (2002) employed choice functions by which appropriate low level heuristics (moving strategies) can be chosen and combined adaptively to search the solutions. Good results on project presentation problems were presented compared with the solutions generated by a random hyper-heuristic. A distributed choice function method was proposed by Gaw, Rattadilok and Kwan (2004) within a hyper-heuristic for timetabling and scheduling problems.

Burke, Kendall and Soubeiga (2003) employed a Tabu Search as the high level heuristic to search through a space of moving strategies for university course timetabling and nurse rostering problems. Good results on both of the problems tested indicated the generality and efficiency of the hyper-heuristic approach. This approach is adopted and extended in (Burke, Landa Silva and Soubeiga 2005) with the aim of investigating the learning of low level heuristics that are suitable and effective for individual objectives in multiple-objective space allocation and course timetabling problems. Promising results are obtained compared with the state-of-the-art approaches. A genetic algorithm methodology was employed as the high level approach by Han and Kendall (2003) and was tested on simulated course scheduling and student project presentation problems. Dowsland, Soubeiga and Burke (2005) introduced the Tabu Search hyper-heuristic, which was investigated in (Burke, Kendall and Soubeiga 2003), within a Simulated Annealing in search of a set of low level heuristics (both neighborhood structures and sampling policies within the solution space) to determine the shipper sizes in transportation problems.

Burke, Petrovic and Qu (2006) employed a Case-Based Reasoning methodology as a heuristic selector for solving course timetabling problems. Problem information is modeled with the corresponding *good* meta-heuristics and stored in the Case-Based Reasoning system. The new problems are solved by using the suggested meta-heuristics which worked well on solving previous similar problems.

1.3.2 Constructive Techniques within Hyper-heuristics

There are only a few papers which employ constructive techniques as low level heuristics within hyper-heuristics for timetabling and scheduling problems. Terashima-Marin, Ross and Valenzuela-Rendon (1999) investigated employing Genetic Algorithms to evolve the configurations of constraint satisfaction methods on constructing problem solutions. The non-direct representations were suggested for Genetic Algorithms to solve large and complex exam timetabling problems.

Ross et al (2003) investigated a genetic-based hyper-heuristic employing 4 basic constructive heuristics on one dimensional bin packing problems. Optimal or near optimal results have been found and potential research issues were also discussed.

Another technique was studied by Asmuni, Burke and Garibaldi (2004) who employed fuzzy rules to combine graph heuristics to construct exam timetables.

Burke, Petrovic and Qu (2006) employed Case-Based Reasoning as a heuristic selector for solving exam timetabling problems. Problem solving situations and the corresponding constructive heuristics were stored in the Case-Based Reasoning system. Solutions for new problems were constructed by repeatedly using the constructive heuristics suggested by the system. Benchmark exam timetabling problems were tested and the results were within the range of those generated by using the state-of-the-art approaches.

The graph based hyper-heuristic (GHH) described in this paper is a constructive approach that employs different graph heuristics during the process of constructing the solution according to the different problem solving situations that might occur at particular times. The next section presents the GHH approach upon graph heuristics and a random ordering method. Some fundamental issues concerning the search space and solution space are also addressed. The investigation and experimental study of GHH and of a multi-stage GHH that are developed for both exam and course timetabling problems are presented in section 3. Our conclusions and potential extensions of this work are presented in the final section.

2. The Graph Based Hyper-heuristic (GHH) Approach

2.1 Graph Heuristics

Graph heuristics are widely studied methods which were developed during the early days of research on timetabling problems (e.g. Welsh and Powell, 1967, Brelaz, 1979). For an introduction to such approaches see (Burke, Kingston and de Werra 2004). They are used in sequential (or constructive) solution methods to order the events that are not yet scheduled according to the difficulties of scheduling them into a feasible timeslot (without violating any hard constraints). The difficulties are represented by the degrees of the vertices in the graph, which model the timetabling problem by representing the events as vertices and conflicts by edges. We say two events in timetabling problems have a *conflict* if they involve the same students. The difficulty of an event is represented by the number of conflicts it has with the others. The objective is to construct a timetable by scheduling the most conflicting events one by one into feasible timeslots, satisfying as many of the soft constraints as possible.

Within our graph-based hyper-heuristic, we employ graph heuristics to schedule a number of events at each step during the construction of the solution. The work presented in this paper investigates the following 5 low level heuristics, with and without randomness (introduced by a random ordering method):

- least Saturation Degree first (SD). Events are ordered (in an increasing order) in terms of the number of feasible timeslots available in the partial solution at that time. The priorities (degrees) of events to be ordered and scheduled are changed dynamically as the solution is constructed.
- largest Color Degree first (CD). Events are ordered (in a decreasing order) in terms of the number of conflicts (events with common students involved) that they have with those already scheduled in the timetable. The degrees of the events not yet scheduled are changed according to the situations encountered at each step of the solution construction.
- Largest Degree first (LD). Events are ordered decreasingly by the number of conflicts they have with other events. This heuristic aims to schedule first those events which have the most conflicts.
- Largest Enrollment first (LE). Events are ordered (in a decreasing order) by the number of students enrolled. This heuristic schedules first those events with the largest numbers of students.
- Largest Weighted Degree first (LWD). Events are ordered (in a decreasing order) by the number of conflicts, each of which is weighted by the number of students involved.

Among events with the same degree, this heuristic gives higher priority to those with a larger number of students involved.

- Random Ordering (RO). Events that are not yet scheduled are ordered randomly. This basic heuristic brings some randomness into the scheduling, which sometimes produces better results in timetabling problems as it increases the exploration of the search space.

2.2 The Graph Based Hyper-heuristic Framework

In the hyper-heuristic framework that we present in this paper, Tabu Search is employed to search for the list of low level heuristics (permutations of graph heuristics and a random ordering method), which are used to construct the complete solutions for timetabling problems. Tabu Search has been widely employed for solving complex scheduling and optimization problems (Glover and Kochenberger, 2003). The basic approach searches the search space of the solutions by iteratively moving to the best neighborhoods of the current solution, whilst keeping a record of recently visited solutions which it cannot re-visit again for a certain number of steps (known as the tabu tenure).

Figure 1 presents the pseudo-code of the Tabu Search within the hyper-heuristic approach we have developed. The search space of the Tabu Search (as the high level heuristic) consists of all of the possible permutations of the low level heuristics described in section 2.1. A move in Tabu Search is to pick a new heuristic list that is obtained by randomly changing two of the heuristics in the previous heuristic list. The newly visited heuristic lists are added into the tabu list (which has a length of 9. i.e. the tabu tenure is 9). The determination of this value is based on our experiments and upon the value recommended from the literature (Reeves, 1996). of course, what this means is that when the Tabu Search selects a heuristic list, that list cannot be re-visited within 9 steps of the search. The objective of the Tabu Search is to find the heuristic list that generates the best quality solution for the timetabling problem under consideration. The search process of Tabu Search within the hyper-heuristic approach stops after a pre-defined number of iterations (*i* in Figure 1), which we adapt according to the problem size. We set it as 5 times the number of events to keep the computational time low. Of course it is possible to perform more iterations if computational time is not a key issue in the problem being solved.

Each heuristic list selected by a move of the Tabu Search is used to construct a complete solution, whose penalty is fed back to the Tabu Search as a guide to the search in the following stages. As each heuristic in the heuristic list is employed to schedule a number (*e* in Figure 1) of events into the timetable (the reason for doing this is explained below), the length of the heuristic list (*k* in Figure 1) is then set as (number of events)/*e*.

INSERT FIGURE 1 SOMEWHERE HERE.

To reduce the size of the search space of the hyper-heuristic and thus reduce its computational time, we added three mechanisms:

1) As the low level heuristics are graph heuristics which are used to construct the solutions, a list of them may not guarantee to generate a feasible solution (see more detail in Section 2.3). To reduce the time spent on implementing heuristic lists that generate unfeasible solutions, we construct a 'failed list', which represents a *flat memory*, to store the parts of heuristic lists that generate unfeasible schedules during the solution construction (in the hyper-heuristic approach). The initial "failed list" is set as empty and is updated after each step of the Tabu Search to store any new heuristic lists that cannot generate feasible solutions. When a new heuristic list is selected by a move of Tabu Search in the hyper-heuristic, it will be checked (before it is applied to construct the solution) to see if it matches those stored in the 'failed list' that led to unfeasible schedules. For example, if a part of the heuristic list '*h1h2h3..*' (where *h1*, *h2* and h3 are low level heuristics) is stored in the 'failed list' because *h3* cannot generate a feasible schedule at that step, all of the heuristic lists selected later such as '*h1h2h3h4 ..*' or '*h1h2h3h5..*' can be ignored before being applied to construct a solution. This mechanism cuts away a large section of the search space by ignoring the

non-valid heuristic lists before applying them to solve the problems and reduces significantly the computational time of the hyper-heuristic.

2) At each step of the solution construction employing a list of heuristics, a number of events (set to 2 in our experiments because our initial testing indicated that this was an appropriate value) are scheduled by the current heuristic in the list (rather than scheduling just one event with the heuristic). This is motivated by the observation by Burke, Petrovic and Qu (2006) that, when scheduling one event by a heuristic at each step of the solution construction, the heuristics in the best heuristic list found by the hyper-heuristic stay the same after a certain number of steps of scheduling. That is, the best heuristic lists found tend to appear in the form of '*h2h2h2…h2h1h1… h1…*'. Scheduling a number of events at each step reduces the size of the search space of the hyper-heuristic without losing much quality in the solutions generated.

3) The initial heuristic list of the hyper-heuristic approach contains only the Saturation Degree heuristic, in order to have a good starting point in the hyper-heuristic. This is because the Saturation Degree heuristic orders the events not yet scheduled according to the number of available timeslots, which changes dynamically during the search. It is potentially (and experimentally tested to be) more effective (more often) than the other heuristics that order the events in static lists. It is expected that the density of the appearance of Saturation Degree in the best heuristic list will be higher than that of the other low level heuristics.

2.3 Fundamental Issues within the Hyper-heuristic Approach

Hyper-heuristics have been attracting recent attention in timetabling research (see Burke and Petrovic, 2002, Burke, Kendall et al. 2003, Petrovic and Burke, 2004). However, some fundamental underlying issues have not been addressed in depth in the literature. Before presenting the analysis of our experiments, we would like to discuss some of these issues.

As mentioned before, hyper-heuristics operate at a *higher level* than most meta-heuristic approaches in the literature. They operate on heuristics rather than directly on the solutions by indirectly choosing a certain low level heuristic which then operates on the events (which either moves the events in the timetable if the low level heuristics are moving strategies, or assigns the events chosen to timeslots if the low level heuristics are constructive heuristics – which is the approach studied here). Thus, it is necessary to distinguish between the search space of the hyper-heuristic and the solution space of the problem.

Figure 2 presents the relationship between the search space of hyper-heuristics and the solution space of the problem. Note that the search space of the *hyper-heuristic* (on the left side of Figure 2) represents a space of heuristics and the solution space of the *problem* (on the right side of Figure 2) represents a space of potential actual solutions (timetables). The heuristics in the search space of the hyper-heuristic correspond to certain solutions of the problem. Of course, the heuristics (which are constructive heuristics studied here) selected by a move in the high level searching method may not guarantee the construction of a feasible solution. This is because the moves in the hyper-heuristic search concern the change of heuristics and not the actual assignment of the events. For example, in Figure 2 we can see that heuristic *A* moves to heuristic *B*, or heuristic *A* moves to heuristic *C* (within the search space of hyper-heuristic). The corresponding solutions (*b* or *c* in the solution space that are constructed by *B* or *C*) in the solution space are not guaranteed to be feasible. The search space of the hyper-heuristic (which consists of permutations of heuristics) is very large, with a large number of heuristics that generate unfeasible solutions in the solution space of the problem. In the figure, *A* and *C* generate feasible solutions, while B generates an infeasible solution.

INSERT FIGURE 2 SOMEWHERE HERE.

In Figure 2 solutions *a* and *b* (which correspond to heuristics *A* and *B* that are in the same neighborhood in the search space of the heuristics), may not be in the same neighborhoods in the search space of solutions. This gives the hyper-heuristic the ability of *jumping* (not moving) within the problem solution space by moving among the neighborhoods defined at a higher level of search in the heuristic search space. In Figure 2, neighborhood moves are represented as solid arrows and *jumping* moves are represented as dashed arrows. The correspondences between the heuristics and actual solutions are represented by dotted lines.

Also note that in Figure 2, the solution space of the *problem* consists of all of the possible solutions that may be obtained by neighborhoods moves, while they may not correspond to any heuristic list in the search space of the hyper-heuristic. For example solution *d* may be within the neighborhood of solution *a* in the search space of solutions. However *d* may not have a corresponding heuristic in the search space of heuristics. Based upon the above observations, we propose a hypothesis here: the hyper-heuristic (which operates at a higher level of problem solving - solving the problems indirectly) may not *be able to* reach all of the solutions in the solution space of the problem.

A deepest descent local search method is employed within the hyper-heuristic after each move in the high level search. The deepest descent search tries to move the events to other timeslots in the timetable that is generated by a heuristic (searched for by the high level heuristic), aiming at improving the quality of the timetable as quickly as possible. This process terminates as soon as no events can be moved to improve the timetable. The high level heuristic will then make a move to another heuristic, which is used to construct another timetable. An illustrative example is presented in Figure 3 in conjunction with the example shown in Figure 2, where solution *d* in the search space of actual solutions might not corresponds to any heuristic in the search space of heuristic lists, while it might be in a neighborhood of solution *a* and thus may be visited by the deepest descent local search.

INSERT FIGURE 3 SOMEWHERE HERE.

The deepest descent local search method is a simple but robust method, which does not introduce extra domain knowledge within the hyper-heuristic framework. The motivation for this is twofold: Firstly, the deepest descent in the search space of solutions tries to exploit the local areas (bringing the solutions obtained to their local optimum quickly), whilst the high level heuristic in the hyper-heuristic tries to explore the search space; Secondly, the GHH thus is potentially able to explore more of the possible solutions for the problem.

3. GHH on Exam and Course Timetabling Problems

3.1 Real-World Exam Timetabling Problems

We investigate our hyper-heuristic approach upon a set of graph heuristics by applying it to the benchmark exam timetabling problems that are presented in (Carter, Laporte and Lee, 1996). These are real-world problems that have been tested by many state-of-the-art approaches. The size of the problems ranges from 81 to 682 exams and from 611 to 18419 students. The density of the conflict matrix, which gives the ratio of the number of conflicting exams over the overall number of exams, ranges from 0.06 to 0.42. The characteristics of the problems are presented in Table 1.

INSERT TABLE 1 SOMEWHERE HERE.

The hard constraints considered in these problems are represented by the "conflicts" of scheduling two exams with common students into the same timeslot. The soft constraint is concerned with spreading out the students' exams over the timetable so that students will not have to sit exams that are too close to each other. The objective, which is the same as that presented in (Carter, Laporte and Lee, 1996), is to schedule all of the exams into the timeslots, while minimizing the cost on the violations of the soft constraint per student. The objective function of GHH which calculates the cost of violations C(t) within solution t is presented in formula (1) below:

$$\mathrm{C}\,(t) = \left(\sum_{s=0}^{4} w_s \times N_s\right) / S \qquad (1)$$

where

$w_s = 2^s$, s = 0, 1, 2, 3, 4, is the weight that represents the importance of scheduling exams with common students either 4, 3, 2, 1, or 0 timeslots away in timetable *t*.

N_s, s = 0, 1, 2, 3, 4, is the number of students involved in the violation of the soft constraint.

S is the number of students in the problem.

The lower the cost, C(*t*), the better the timetable is. We do not consider any infeasible solutions (i.e. those with violations of hard constraints).

3.1.1 A Graph Based Hyper-heuristic upon a Different Number of Low Level Heuristics

We investigate here the effect of different low level heuristics on the behavior of the GHH for exam timetabling problems. This helps us to gain a deeper understanding about *general* hyper-heuristics which are applicable for a wider range of problems. The combinations of low level heuristics employed are based on the Saturation Degree heuristic with a different number of other heuristics with and without randomness. For all of the 11 problems presented in (Carter, Laporte and Lee, 1996), 3 runs with distinct seeds are carried out and the costs of the best solutions are presented in Table 2.

INSERT TABLE 2 SOMEWHERE HERE.

The values shown under the column “SL” and “SLR”, “SCL” and “SCLR”, and “SCLx” and “SCLxR” in Table 2 are the costs of solutions obtained by GHH upon two, three and all graph heuristics, with and without Random Ordering.

We assume that the larger the number of low level heuristics employed in the GHH, the larger the size of the search space (of the GHH) will be. This implies that it is possible to indirectly explore a larger part of the solution space of the problem (possibly containing more and better results). For example, the search space of GHH upon SCLxR should include the search space of GHH upon SCLR. This is apparent from Table 2, where the best results (in bold) are mostly obtained by GHH with a larger number of low level heuristics.

Although employing a larger number of low level heuristics in GHH tends to obtain a better performance, this is not always the case. For example, GHH upon “SCLx” outperforms “SCLxR”, which employs a larger number of low level heuristics, on 10 out of 11 problems. The reason that GHH, when employing a smaller number of low level heuristics, sometimes outperforms the approach with a larger number of low level heuristics is precisely because the latter has a much larger search space. Some parts of the larger search space may not be explored within the same search time (i.e. number of iterations). To clarify this issue we carried out another set of experiments on “SCLxR” with a higher number of iterations (10 * number of events). The results obtained are presented in the last column entitled “SCLxR(*10)” in Table 2. We can observe that GHH upon “SCLxR” obtained better results (on 5 out of 11 problems than that of “SCLx”) by being given more searching time. As the search space of GHH upon “SCLx” is a subset of that of GHH upon “SCLxR”, it is possible to say that, given enough search time, the results obtained by GHH upon “SCLxR” will be at least not worse than that of GHH upon “SCLx”.

We can also see that GHH upon “SL” and “SCL” with random ordering outperforms the same GHH without randomness, which is not the case for GHH upon “SCLx”. When more searching time is given, GHH upon “SCLxR” performs much better, meaning that GHH employing heuristics with random ordering performs better than without randomness. We may then conclude, from the above observations, that the larger the number of low level heuristics in GHH, the better it may perform, provided a large enough amount of search time is given (but this is an important proviso).

3.1.2 Multi-stage GHH

The observations above provide the motivation for proposing a multi-stage GHH which employs a different number of heuristics in two stages. The multi-stage GHH employs “SCLR” in the first stage and “SCLxR” in the second stage to explore its search space as much as possible (based on the heuristic list obtained in the first stage). The reason for employing “SCLR” in the first stage of GHH is not only because it performs best, but also because it employs the four distinct low level heuristics (while the LE, LD and LWD are based on the same heuristic and may contribute similarly to LD).

The results obtained for the multi-stage GHH are presented in Table 3, together with the best results of the single-stage GHH in the last sub-section. We also present the state-of-the-art approaches reported in the literature (Carter, Laporte and Lee, 1996, Di Gaspero and Schaerf, 2000, Caramia, Dell'Olmo, and Italiano, 2001, Burke and Newall, 2002, Merlot et al, 2002, Asmuni, Abdullah et al, 2004, Burke and Garibaldi, 2004). They are reported as the best results obtained by the corresponding approaches. We have not included an entry for the corresponding computational time because they are not reported in several of these papers.

INSERT TABLE 3 SOMEWHERE HERE.

We can observe that the multi-stage GHH does not outperform the single-stage hyper-heuristic presented in the last sub-section, although it obtains results that are not worse than that of GHH upon "SCLR". The reason may be that it starts from a smaller search space and this may limit the search towards a certain region of the search space.

3.1.3 Summarizing Comments Concerning GHH for Exam Timetabling Problems

For all of the problems tested, GHH upon graph heuristics and random ordering obtained very good results that are within the range of the best results reported in the literature. By searching (on a higher level) the heuristic space, the GHH is capable of *jumping* (not moving) among the solution space of the problem by moving among the heuristic lists in its search space. We believe that this makes it a very efficient approach on difficult problems that have complex, especially disjointed, solution spaces.

Also note that except for (Carter, Laporte and Lee, 1996) where a number of different constructive methods with backtracking were employed, and (Asmuni, Burke and Garibaldi, 2004) where constructing, un-scheduling and rescheduling are performed, all of the other approaches operate by improving on the initial complete solutions obtained beforehand. For the complex problems tested here, it is observed that no feasible solutions can be obtained by the use of pure constructive graph heuristics. In contrast, the proposed hyper-heuristic method is constructive, starting from an empty solution. It is not dependent on initial solutions, which may sometimes affect the behavior of the other approaches. Most importantly, it is also effective for course timetabling problems (see below). Yet, the proposed method obtained competitive results with all state-of-the-art approaches for exam timetabling problems.

We have found that deepest descent local search after each move of the Tabu Search within the GHH approach often improves the solutions obtained and occasionally obtains a better final solution than that obtained by the Tabu Search alone. This may indicate that, during the problem solving, GHH sometimes reaches a point in the solution space (such as solution *a* in Figure 2 and Figure 3) and cannot go further to its local optimal (such as solution *d* in Figure 2 and Figure 3) unless moves upon the actual solutions are made. However, it is difficult to check this hypothesis thoroughly, due to the size of the search space for the solutions of complex timetabling problems. This is because the hypothesis concerns the coverage of the solution space by the search space of the hyper-heuristic (see Section 2.2).

3.2 University Course Timetabling Problems

The proposed GHH method was also tested on eleven benchmark course timetabling problems, proposed by the Metaheuristic Network[1]. This problem description is taken from (Socha, Knowles and Sampels, 2002). The problems[2] need to assign 100-400 courses into a timetable of 5 days, 9 timeslots a day, while satisfying room features and capacity constraints. They are grouped into small, medium and large problems. The hard constraints that must be satisfied are:

1. no students can be scheduled to more than one event at a time
2. the room meets all features required by the event

[1] http://www.metaheuristics.net/
[2] http://iridia.ulb.ac.be/~msampels/ttmn.data/

3. room capacity is respected
4. no more than one event is allowed per room and per timeslot

Soft constraint violations are equally weighted and summed up within the cost function. They are presented below:

1. a student has a class in the last timeslot of the day
2. a student has more than two classes in a row
3. a student has only one class on a day

3.2.1 GHH upon All Low Level Heuristics

We employ exactly the same GHH approach with the same low level heuristics tested on the exam timetabling problems. The only changes made, in order to deal with these different problems, are the cost function which also evaluates the room constraints. The best results by 5 runs of GHH with distinct seeds upon all of the low level heuristics are presented in Table 4, which also presents the results of 3 approaches (Socha, Knowles and Sampels, 2002, and Burke, Kendall and Soubeiga, 2003) reported in the literature for comparison. For the "Hyper-heuristic" the best results out of 5 runs obtained are presented, for the "Local Search" and "Ant Algorithm" the average results out of 50 runs on small problems, 40 runs on medium problems and 10 runs on large problems are reported. We test our approach with the same number of runs as that of the "Hyper-heuristic" from (Burke, Kendall and Soubeiga, 2003) to make a more fair comparison. The term "x% Inf" in Table 4 indicates the percentage of runs which failed to obtain feasible solutions.

From Table 4 we can observe that our GHH approach obtains competitive results with the other 3 approaches on these course timetabling problems. For problem "Medium5", it obtained the best results among all approaches. It outperforms the "Local Search" method on all of the problems except "Medium1", "Medium2" and "Medium4". And for all of the problems tested it finds feasible solutions with all the 5 distinct seeds tested, which the "Tabu

Search Hyper-heuristic" and "Local Search" approaches failed to obtain.

INSERT TABLE 4 SOMEWHERE HERE.

3.2.2 Summarizing Comments Concerning GHH on Course Timetabling Problems

The experimental results on GHH for course timetabling problems demonstrate its ability to work well on hard problems. We can observe that for problems "Medium5" and "Large", which are the hardest (highest costs obtained by the other approaches), GHH obtained the best and second-best results. The reason may be that the GHH is capable of *jumping* (while not *moving*) within the solution space by the moves within the search space of heuristic lists. This enables it to deal well with a broad range of hard problems (with a complex, and probably disjoint, solution space).

The deepest descent local search after each move of the Tabu Search improves the solution in most cases during the search of GHH. The best final results are sometimes from the improvement made by deepest descent local search on the solutions obtained by Tabu Search. This may strengthen our hypothesis about the coverage of the solution space vs. search space within our GHH approach.

For all the other approaches compared, initial solutions are required before the algorithms are performed. Our GHH solves the problem by starting from an empty solution in each move of Tabu Search. For the course timetabling problems tested here, the violations of soft constraints 2 (more than 2 consecutive courses) and 3 (single course assigned in one day) cannot be evaluated accurately until a complete solution is obtained. This is not true for the other approaches compared here, since complete solutions exist. In our GHH approach the evaluation can only be made approximately during the construction of the solution. This may limit the search of GHH when searching for globally optimal solutions, although the deepest descent local search method upon the complete solution after each step of Tabu Search can improve the timetables concerning these soft constraints.

4. Conclusions

The overall goal of this paper was to investigate a hyper-heuristic which operates at a higher level of generality than most of the current approaches studied in timetabling. Current state-of-the-art approaches are the result of a significant investment of effort in developing sophisticated and elaborate heuristics, which are "tailor made" for their particular problems. In our hyper-heuristic framework both the Tabu Search and graph heuristics are general methods that are widely applicable. We have presented a general constructive approach which is not dependent on the initial solution that the other approaches need to generate.

By employing simple and general heuristics in an intelligent way, the hyper-heuristic is capable of generating comparable results to those of special purpose approaches. The hyper-heuristic has the ability of selecting general heuristics, picking up the events that are most *difficult* to schedule (by different heuristics), in any given solution construction state. Although the goal of the present study is not to beat the specific approaches in the literature, the GHH works well on all of the problems and for one of the benchmark course timetabling problems, the hyper-heuristic we developed actually obtained the best result among all those reported in the literature. It is a simple, robust and very effective general approach that does not use any domain knowledge except in the cost function that deals with different constraints in different problems.

Experimental results indicate that the hyper-heuristic works more efficiently when using a larger number of low level heuristics. However, the size of the search space of GHH increases as well, which also increases the computational time. The multi-stage GHH was studied with the aim of getting good results in the first stage of GHH employing less low level heuristics and improving the results in the second stage employing more low level heuristics based on the heuristic lists obtained from the first stage. However the experimental results presented were worse than the single stage GHH due to the limitation of the starting points for the search of GHH.

Our hypothesis on the coverage of the solution space of the problem by searching within the search space of the hyper-heuristic is experimentally strengthened for both the exam and course timetabling problems. This hypothesis needs to be tested on a larger range of timetabling and scheduling problems.

5. Future Work

Future work could use other simple heuristics or approaches, instead of Tabu Search, in exploring the search space of heuristics. A larger number of low level general heuristics can also be added to the hyper-heuristic framework to explore a larger section of the solution space of the problems. It might also be interesting to consider more than one low level heuristic when choosing the events to be scheduled at each step of the solution construction. Combining different low level heuristics at a single step of the solution construction may find more appropriate events to be scheduled. Due to the large search space of the hyper-heuristic, future work will also need to investigate the impact of additional low level heuristics on computational time.

The hyper-heuristic approach described here represents a general and simple framework equipped with little domain knowledge, and may be easily applied to many other timetabling and scheduling problems with little effort. Future work should extend the same framework to other problems.

Acknowledgements

We would like to acknowledge EPSRC for sponsorship of this research (grant number GR/N36837/01). Professor Meisels's contribution was supported by an EPSRC Visiting Fellowship (GR/S53459/01). We would also like to thank the anonymous referees for their helpful comments.

References

Abdullah S., Ahmadi S, Burke E.K. and Dror M. 2004. Investigating Ahuja-Orlin's Large Neighbourhood Search for Examination Timetabling. Technical Report NOTTCS-TR-2004-8, School of CSiT, University of Nottingham, U.K.

Abramson D., Krishnamoorthy M. and Dang H. 1999. Simulated Annealing Cooling Schedules for the School Timetabling Problem. Asia-Pacific Journal of Operational Research, 16 1-22.

Asmuni H., Burke E.K. and Garibaldi J. 2004. 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, to appear in Lecture Notes in Computer Science.

Banks D., Beek P. and Meisles A. 1998. A Heuristic Incremental Modelling Approach to Course Timetabling. Proceedings of the 12th Canadian Conference on Artificial Intelligence, pp. 16–29.

Bardadym V. 1996. Computer-Aided School and University Timetabling: The New Wave. 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, pp. 22-45.

Beddoe G. and Petrovic, S. 2005. Determining feature weights using a genetic algorithm in a case-based reasoning approach to personnel rostering. Accepted for publication in European Journal of Operational Research, 2005.

Brelaz D. 1979. New Methods to Color the Vertices of a Graph. Communications of the ACM, 22(4) 251-256.

Burke E.K., Bykov Y., Newall J. and Petrovic S. 2004. A Time-Predefined Local Search Approach to Exam Timetabling Problems. IIE Transactions on Operations Engineering, 36(6) 509-528.

Burke E.K. and Carter M. eds. 1998. Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1408.

Burke E.K. and De Causmaecker P. (eds.) 2003. Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2740.

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

Burke E.K., Dror M., Petrovic S. and Qu R. 2005. Hybrid Graph Heuristics within a Hyper-heuristic Approach to Exam Timetabling Problems. B.L. Golden, S. Raghavan and E.A. Wasil (eds.). The Next Wave in Computing, Optimization, and Decision Technologies. Springer. pp. 79-91.

Burke E.K. and Erben W. (eds.) 2001. Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2079.

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

Burke E.K., Kendall G., Newall J., Hart E., Ross P. and Schulenburg, S. 2003. Hyper-heuristics: an Emerging Direction in Modern Search Technology. In: Glover F. and Kochenberger G. (eds.) Handbook of Meta-Heuristics, Kluwer, pp. 457-474.

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

Burke E.K., Kingston J. and de Werra D. 2004. Applications to Timetabling, In: Gross, J. and Yellen, J. (eds.), Handbook of Graph Theory. Chapman Hall/CRC Press. pp. 445-474.

Burke E.K., Landa Silva J.D. and Soubeiga E. 2005. Multi-objective Hyper-heuristic Approaches for Space Allocation and Timetabling. To appear in: Ibaraki T., Nonobe K. and Yagiura M. (eds.). Meta-heuristics: Progress as Real Problem Solvers. Springer.

Burke E.K., MacCarthy B., Petrovic S. and Qu R. 2000. Structured Cases in Case-Based Reasoning - Re-using and Adapting Cases for Time-tabling Problems. Knowledge-Based Systems, 13(2-3) 159-165.

Burke E.K., MacCarthy B., Petrovic S. and Qu R. 2001. Case-based Reasoning in Course Timetabling: An Attribute Graph Approach. In: David A. and Watson I. (eds.), Case-Based Reasoning Research and Development, Lecture Notes in Artificial Intelligence 2080. pp. 90-104.

Burke E.K., MacCarthy B., Petrovic S. and Qu R. 2003. Knowledge Discovery in Hyper-heuristic Using Case-Based Reasoning on Course Timetabling. In: Burke E.K. and De Causmaecker P. (eds.) Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2740. pp. 90-103.

Burke E.K., MacCarthy B., Petrovic S. and Qu R. 2005. Multiple-Retrieval Case-Based Reasoning for Course Timetabling Problems. To appear in Journal of Operations Research Society, 2005.

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

Burke E.K. and Newall J. 2002. Enhancing Timetable Solutions with Local Search Methods. In: Burke E.K. and De Causmaecker P. (eds.) Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2740. pp. 195-206.

Burke E.K., Newall J. and Weare, R. 1996. A Memetic Algorithm for University 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, pp. 241-250.

Burke E.K., Newall J. and Weare R. 1998. Initialization Strategies and Diversity in Evolutionary Timetabling. Evolutionary Computation, 6(1) 81-103.

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

Burke E.K., Petrovic S. and Qu R. 2006. Case Based Heuristic Selection for Timetabling Problems. To appear in Journal of Scheduling 2006.

Burke E.K. and Ross P. (eds.) 1996. Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1153.

Burke E.K. and Trick M. (eds.) 2004. Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling, Pittsburg PA, U.S.A. August 2004. To appear in Lecture Notes in Computer Science, 2005.

Caramia M. Dell'Olmo P. and Italiano G. 2001. New Algorithms for Examination Timetabling. In: S. Naher and D. Wagner (eds.) Algorithm Engineering 4th International Workshop, WAE 2000. Lecture Notes in Computer Science 1982, pp. 230-241.

Carter M. 1986. A Lagrangian Relaxation Approach to the Classroom Assignment Problem. INFOR 27(2) 230-246.

Carter M. and Laporte G. 1996. 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, pp. 3-21.

Carter M. and Laporte G. 1998. Recent Developments in Practical Course Timetabling, In: Burke E.K. and Ross P. (eds.), Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1408, pp. 3-19.

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

Casey S. and Thompson J. 2002. GRASPing the Examination Scheduling Problem. In: Burke E.K. and De Causmaecker P. (eds.) Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2740. pp. 232-246.

Costa D. 1994. A Tabu Search for Computing an Operational Timetable. European Journal of Operational Research. 76 98-110.

Cheang B., Li H., Lim A. and Rodrigues B. 2003. Nurse Rostering Problems: A Bibliographic Survey. European Journal of Operational Research, 151(3) 447-460.

Côté P., Wong T. and Sabourin R. 2005. Application of a Hybrid Multi-Objective Evolutionary Algorithm to the Uncapacitated Exam Proximity Problem. In: Burke E.K. and Trick M. (eds.) Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling , to appear in Lecture Notes in Computer Science, Springer, 2005.

de Werra D. 1985. Graphs, Hypergraphs and Timetabling. Methods of Operations Research. 49 201-213.

Deris B., Omatu S., Ohta H. and Samat D. 1997. University Timetabling by Constraint-based Reasoning: A Case Study. Journal of Operational Research Society. 48(12) 1178-1190.

Dowsland K. 1998. Off the Peg or Made to Measure. In: Burke E.K. and Carter M. (eds.) Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1408. pp. 37-52.

ten Eikelder H.M.M. and Willemen RJ. 2000. Some Complexity Aspects of Secondary School Timetabling Problems. In: Burke E.K., Erben W. (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 3rd International Conference. Lecture Notes in Computer Science 2079, pp. 18-27.

Erben W. 2000. A Grouping Genetic Algorithm for Graph Coloring and Exam 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. pp. 132-158.

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

Di Gaspero L. and Schaerf A. 2000. 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. pp. 104-117.

Dowsland K., Soubeiga E. and Burke E.K. 2005. A Simulated Annealing Hyper-heuristic for Determining Shipper Sizes. Accepted by the European Journal of Operational Research, 2005.

Gaw A., Rattadilok P. and Kwan R.S. 2004. Distributed Choice Function Hyperheuristics for Timetabling and Scheduling. In: Burke E.K. and Trick M. (eds.): In: Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling, to appear in Lecture Notes in Computer Science, 2005.

Glover F. and Kochenberger G. 2003. Handbook of Metaheuristics, Kluwer.

Han L., Kendall G. 2003. Investigation of a Tabu Assisted Hyper-Heuristic Genetic Algorithm. Congress on Evolutionary Computation, Canberra, Australia, pp. 2230-2237.

Kendall G., Cowling P. and Soubeiga E. 2002. Choice Function and Random HyperHeuristics. Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, pp. 667-671.

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

Leake D. (ed.) 1996. Case-based Reasoning: Experiences, Lessons and Future Directions. AAAI Press, Menlo Park, CA.

Lewis, R. and Paechter, B. 2004. New Crossover Operators for Timetabling with Evolutionary Algorithms. 5th International Conference on Recent Advances in Soft Computing, Nottingham, UK, Volume 5, pp. 189-195.

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

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

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

Qu R. and Burke E.K. 2005. Hybrid Variable Neighborhood Hyper-heuristics for Exam Timetabling Problems. Accepted by Meta-heuristic International Conference. Vienna, Aug 2005.

Reeves C.R. 1996. Modern Heuristic Techniques. In: R-Smith V.J., Osman I.H., Reeves C.R. and Smith G.D. (eds.) Modern Heuristic Search Methods, pp. 1-25.

Ross P., Marin Blasques J.G., Schulenburg S. and Hart E. 2003. Learning a Procedure that Can Solve Hard Bin-Packing Problems: A New GA-Based Approach to Hyper-heuristics. In Cantú-Paz E. et al (eds.), Genetic and Evolutionary Computation 2003, Lecture Notes in Computer Science 2724, pp. 1295-1306.

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

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

Terashima-Marin H., Ross P. and Valenzuela-Rendon M. 1999. Evolution of Constraint Satisfaction Strategies in Examination Timetabling. Genetic Algorithms and Classifier Systems 1999. pp. 635-642.

Welsh D.J.A. and Powell M.B. 1967. The upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal. 11 41-47.

```
initial heuristic list $hl$ = {$h_1$ $h_2$ $h_3$ … $h_k$}
//Begin of Tabu Search
for $i$ = 0 to $i$ = (5 * the number of events) //number of iterations
  $h$ = change two heuristics in $hl$ //a move in Tabu Search
  if $h$ does not match a heuristic list in ‘failed list’
       if $h$ is not in the tabu list //$h$ is not recently visited
       for $j$ = 0 to $j$ = $k$ //$h$ is used to construct a complete solution
            schedule the first 2 events in the event list ordered using $h_j$
            if no feasible solution can be obtained
               store $h$ into the ‘failed list’ //update “failed list”
            else if cost of solution $c$ < the best cost $c_g$ obtained
               save the best solution, $c_g$ = c //keep the best solution
               add $h$ into the tabu list
               remove the first item from the tabu list if its length > 9
        $hl$ = $h$
  //end if
  Deepest descent on the complete solution obtained
//end of Tabu Search
output the best solution with cost of $c_g$
```

Figure 1 Pseudo-code of Tabu Search within the graph based hyper-heuristic

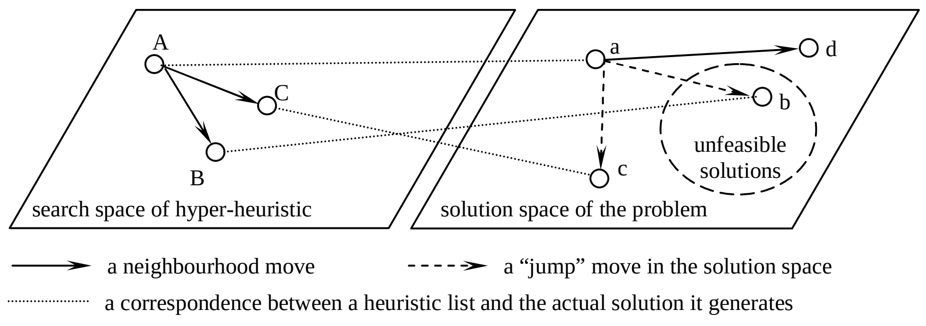
A
C
B
search space of hyper-heuristic
a
d
b
c
unfeasible solutions
solution space of the problem
a neighbourhood move
a “jump” move in the solution space
a correspondence between a heuristic list and the actual solution it generates

Figure 2 The relation between the search space of the hyper-heuristic and solution space of the problem

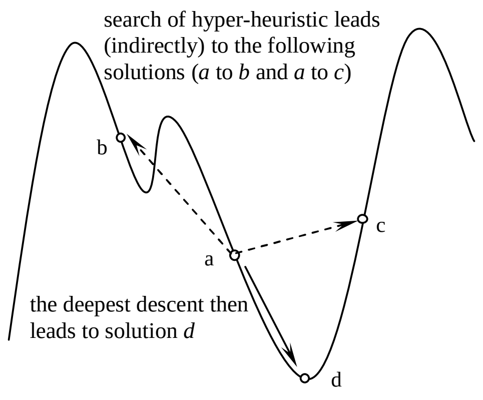
search of hyper-heuristic leads
(indirectly) to the following
solutions (*a* to *b* and *a* to *c*)
b
c
a
the deepest descent then
leads to solution *d*
d

Figure 3 An illustration of how the deepest descent local search could lead to solutions that are not represented by the heuristic lists (searched by the high level search)

Table 1 Characteristics of Benchmark Exam Timetabling Problems

	car91	car92	ear83	hec92	kfu93	lse91	sta83	tre92	ute92	uta93	york83
exams	682	543	190	81	461	381	139	261	184	622	181
students	16925	18419	1125	2823	5349	2726	611	4360	2750	21266	941
timeslots	35	32	24	18	20	18	13	23	10	35	21
matrix density	0.13	0.14	0.27	0.42	0.6	0.6	0.14	0.18	0.8	0.13	0.29

Table 2 Costs of solutions obtained by GHH upon a different number of heuristics (S: saturation degree, L: largest degree, C: color degree; R: random ordering, Lx: largest weighted, largest enrollment and largest degree)

Problem	SL	SLR	SCL	SCLR	SCLx	SCLxR	SCLxR(10*)
car91	5.73	5.55	5.51	***5.41***	5.78	6.13	5.67
car92	5.01	***4.79***	4.89	4.84	4.88	4.93	4.91
ear83	40.54	39.1	40.47	39.17	**38.19**	40.37	40.23
hec92	13.41	12.86	13.03	13.11	13.89	**12.72**	***12.55***
kfu93	16.63	15.83	***15.76***	16.01	15.91	17.03	15.83
lse91	13.71	13.88	14.12	13.46	**13.15**	13.88	***13.11***
sta83	150.22	143.3	148.54	143.75	**141.08**	142.83	142.03
tre92	9.19	9.38	***8.85***	9.27	8.97	9.27	9.15
ute92	32.71	32.01	32.03	32.01	**31.65**	32.1	31.83
uta93	4.07	4.18	4.0	***3.54***	3.75	3.77	3.65
york83	46.21	44.0	45.05	44.51	**40.13**	48.78	46.03

Table 3 Results from the GHH, multi-stage GHH and the best results reported in literature on benchmark exam timetabling problems

	car91	car92	ear83	hec92	kfu93	lse91	sta83	tre92	ute92	uta93	york83
GHH (best)	5.41	4.84	38.19	12.72	15.76	13.15	141.08	8.85	32.01	3.54	40.13
Multi-stage GHH	5.41	4.84	38.84	13.11	15.99	13.43	142.19	9.2	31.65	3.54	44.51
Abdullah et al	5.21	4.36	34.87	10.28	**13.46**	10.24	150.28	**8.13**	**24.21**	3.63	**36.11**
Asmuni et al	5.20	4.52	37.02	11.78	15.81	12.09	160.42	8.67	27.78	3.57	40.66
Burke &Newall $_{2002}$	4.6	**4.0**	37.05	11.54	13.9	10.82	168.73	8.35	25.83	**3.2**	36.8
Burke,Bykov et al $_{2004}$	**4.2**	4.8	35.4	10.8	13.7	10.4	159.1	8.3	25.7	3.4	36.7
Caramia et al	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	158.2	9.4	24.4	3.5	36.2
Casey & Thompson	5.4	4.4	34.8	10.8	14.1	14.7	**134.7**	8.7	25.4	Inf	37.5
Carter et al	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	25.8	3.5	41.7
Di Gapero & Schaerf	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	29.0	4.2	41.0
Merlot et al	5.1	4.3	35.1	10.6	13.5	10.5	157.3	8.4	25.1	3.5	37.4

Table 4 Results of the GHH, multi-stage GHH and the other 3 approaches in literature on benchmark course timetabling problems

	GHH upon 6 heuristics	*Tabu Search Hyper-Heuristic* Burke, Kendall &Soubeiga 2003	*Local Search* Socha, Knowles & Sampels 2002	*Ant Algorithm* Socha, Knowles & Sampels 2002
Small1	6	**1**	8	**1**
Small2	7	**2**	11	3
Small3	3	**0**	8	1
Small4	3	**1**	7	**1**
Small5	4	**0**	5	**0**
Medium1	372	**146**	199	195
Medium2	419	**173**	202.5	184
Medium3	359	267	77.5% Inf	**248**
Medium4	348	169	177.5	**164.5**
Medium5	**171**	303	100% Inf	219.5
Large	1068	80% Inf 1166	100% Inf	**851.5**