Journal of the Operational Research Society (2009) 60, 1273–1285

www.palgrave-journals.com/jors/

Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems

R Qu* and EK Burke

University of Nottingham, Nottingham, UK

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 hybridizations 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.

Journal of the Operational Research Society (2009) **60,** 1273–1285. doi:10.1057/jors.2008.102
Published online 22 October 2008

Keywords: university timetabling; graph colouring heuristics; hyper-heuristics; tabu search; variable neighbourhood search; iterated local search

1. Introduction

1.1. Timetabling problems

Timetabling problems have been widely investigated in the operational research and artificial intelligence research communities for more than four decades (eg see de Werra, 1985; Schaerf, 1999; Burke *et al*, 2004c; Qu *et al*, 2008). They appear in the forms of educational timetabling (eg Bardadym, 1996; Carter and Laporte, 1996, 1998; Burke and Petrovic, 2002; Petrovic and Burke, 2004; Qu *et al*, 2008), nurse rostering (eg Burke *et al*, 2004b), sports timetabling (eg Easton *et al*, 2004) and transportation scheduling (eg Kwan, 2004).

This paper is concerned with university (course and exam) timetabling problems, which have attracted particular attention over the years. They are important administrative activities that 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 (eg de Werra, 1985; Burke *et al*, 1997; Schaerf, 1999; Burke and Petrovic, 2002; Burke *et al*, 2004c, Petrovic and Burke, 2004; Qu *et al*, 2008). Specific papers on course (Carter and Laporte, 1998) and exam (Carter and Laporte, 1996) timetabling provide excellent surveys up to 1998 and 1996, respectively. A recent survey of exam timetabling is presented in Qu *et al* (2008). A discussion of the difference between course and exam timetabling is presented in McCollum (2007).

In Burke *et al* (2004c), 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 set of timeslots while satisfying a number of constraints. Constraints in university timetabling can usually be grouped into two types:

- *Hard constraints* that must be satisfied under any circumstances. For example, conflicting events (ie those courses/exams/meetings with common resources such as students) cannot be scheduled simultaneously, that is, if $D_{e_s e_p}$ is the number of students who are enrolled in both exams e_s and e_p; x_{e_s} is the timeslot to which exam e_s is assigned, $E = \{e_1, e_2, \ldots e_n\}$ is the set of exams, then

$$\forall e_s, e_p \in E, (s \neq p \text{ and } D_{e_s e_p} > 0) \Rightarrow x_{e_s} \neq x_{e_p}$$

- *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.

*Correspondence: R Qu, Automated Scheduling, Optimisation and Planning (ASAP) Group, School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK.
E-mail: rxq@cs.nott.ac.uk

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 (ie with only the conflicting hard constraints) can be modelled as graph colouring problems (Burke *et al*, 2004c). Graph colouring heuristics represent some of the very early timetabling approaches (from the 1960s) (eg de Werra, 1985; Burke *et al*, 2004c). Constraint-based techniques have also been employed (eg White, 2000; Merlot *et al*, 2002) by modelling timetabling problems as Constraint Satisfaction Problems (Brailsford *et al*, 1999). Meta-heuristics (Reeves, 1993; Burke and Kendall, 2005) have been very successful over a range of complex timetabling problems in recent research. These include tabu search (eg Nonobe and Ibaraki, 1998; Di Gaspero and Schaerf, 2001; White *et al*, 2004), simulated annealing (eg Thompson and Dowsland, 1998; Burke *et al*, 2004a; Dowsland *et al*, 2007) and evolutionary algorithms (eg Burke and Newall, 1999; Terashima-Marin *et al*, 1999).

Recent new techniques and methodologies on timetabling include ant algorithms (eg Socha *et al*, 2002; Eley, 2007), case-based reasoning (eg Burke *et al*, 2005, 2006), fuzzy reasoning (eg Asmuni *et al*, 2005), GRASP (eg Casey and Thompson, 2003) and hybrid approaches (Caramia *et al*, 2008). Novel neighbourhood design in search algorithms has also gained some success, that is, very large-scale neighbourhood search (Abdullah *et al*, 2007a, b; Meyers and Orlin, 2007), multi-neighbourhood search (Abdullah *et al*, 2007c) and variable neighbourhood search (Burke *et al*, 2008), etc.

1.2. Hyper-heuristics

Most meta-heuristic (Reeves, 1993; Burke and Kendall, 2005) 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' (Burke *et al*, 2003a; Ross, 2005). 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 (ie those to be selected) in a hyper-heuristic 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, we can 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 from, 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 the 1960s (Crowston *et al*, 1963; Fisher and Thompson, 1963), although, at that time, the particular term was not used (Burke *et al*, 2003a). The basic idea is fundamentally different from most heuristic implementations. For example, in resource constrained project scheduling (Brucker and Knust, 2006), permutations of jobs are usually obtained by using certain (fixed) strategies and jobs are then scheduled one by one in this specific order. In a hyper-heuristic approach, the aim would be to *search* for a *set* of (low level) strategies that 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 Cicirello and Smith (2004), 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 Gomes and Selman (2001), algorithm portfolios were developed where a set of heuristics was used for solving combinatorial optimization problems. In Nareyek (2003), 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 (ie Burke *et al*, 2003b; Gaw *et al*, 2005; Dowsland *et al*, 2007) or on using constructive heuristics (Terashima-Marin *et al*, 1999; Ross *et al*, 2004; Burke *et al*, 2005, 2006, 2007; Bilgin *et al*, 2007) as the low-level heuristics. In Burke *et al*, (2003b), tabu search was employed as the high-level heuristic upon a set of moving strategies on both the nurse rostering and course timetabling problems. A simulated annealing hyper-heuristic was developed in Dowsland *et al* (2007) to deal with the shipper sizes in transportation problems. In Gaw *et al* (2005), 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 Bilgin *et al* (2007), heuristic selection and move acceptance criteria were analysed in depth within a hyper-heuristic for both optimization functions and exam

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

	*car*91 *I*	*car*92 *I*	*ear*83 *I*	*hec*92 *I*	*kfu*93	*lse*91	*sta*83 *I*	*tre*92	*ute*92	*uta*93 *I*	*yok*83 *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	16 925	18 419	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

timetabling problems. In Burke *et al* (2006), 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 Ross *et al* (2004) 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 Terashima-Marin *et al* (1999) for constructing exam timetables.

In our previous work Burke *et al* (2007), 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 a 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. Hybridizations 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 that are quite different in terms of problem constraints (McCollum, 2007). 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, nine 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.

The exam timetabling problems we tested in this paper were first introduced in Carter *et al* (1996), and have been widely tested by a number of approaches during the last 10 years (Qu *et al*, 2008). The data set 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 data sets (and the difficulties caused by different instances circulating under the same name) was given in Qu *et al* (2008). 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 data set. 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. More details can be found at (http://www.asap.cs.nott.ac.uk/resources/data.shtml).

The rest of the paper is organized 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 GHH framework

2.1. Graph colouring heuristics

Graph colouring heuristics (Burke *et al*, 2004c) are constructive heuristics that 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.

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 initialization methods for meta-heuristics (eg Burke *et al*, 2004c), or intelligently hybridized to solve complex timetabling and optimization problems (Asmuni *et al*, 2005; Burke *et al*, 2005, 2007) for exam timetabling. This motivates us to investigate using hyper-heuristics to choose graph colouring heuristics for constructing high-quality timetables.

2.2. The GHH approach

In our previous GHH (Burke *et al*, 2007), 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 first 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

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.

The GHH approach (Burke *et al*, 2007) 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 search is carried out upon combinations of difficulty measuring

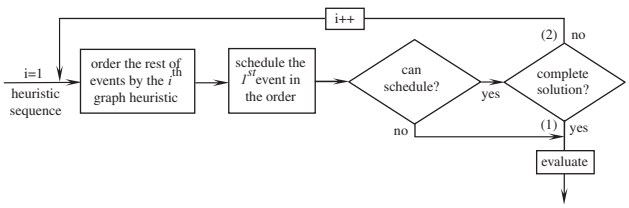

Figure 1 Solution construction by a sequence of graph heuristics.

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

```
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 c_g obtained
              save the best solution, c_g = c
                 update the tabu list
                 hl = h
    //end if
//end of Tabu Search
output the best solution with penalty of c_g
```

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

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 (ie 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.

2.3. 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 optimization problem P, let H be the search space of P, and let f be the objective function that we are aiming to optimize. 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:

- *Optimization problem P*: To optimize 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 to 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 *f(h)*, are allowed in the GHH framework.
- *Low-level heuristics*: Elements (graph colouring heuristics) in the heuristic sequences h, which are used to order events and 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. 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 Burke *et al* (2007).

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 Burke *et al*, 2007). 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 that 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

We test a simple steepest descent method (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

In the iterated local search (ILS) (Lourenco *et al*, 2003), the search is re-started after a certain number of iterations of the 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 the 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

The tabu search (TS) we employ here is similar to that of our previous work (Burke *et al*, 2007). 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 (Reeves, 1993). The search stops after the set number of evaluations is carried out.

3.4. Variable neighbourhood search

The motivation for employing variable neighbourhood search (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 Hansen and Mładenovic, 2001; Hansen and Mladenovic, 2005). 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. Hybridizations within the GHH framework

Based on the GHH framework, we further explore how local search can be hybridized in terms of the search within the two search spaces. We also further investigate the hypothesis in Burke *et al* (2007) 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 hybridizing the local search, which is a simple greedy search, within the GHH (see neighbourhood operator on solutions in Table 4). 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 first 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 first event measured by the next graph colouring heuristic is scheduled based on the improved partial solution from the last step.

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

```
initialisation of the heuristic sequence hl = {h1h2 … 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
          if moving strategy satisfied
          hl = h
          *update tabu list //optional for Tabu Search

//end of high level search
output the best solution with penalty of cg
```

Figure 3 The hybrid GHH framework.

Table 4 Characteristics of the search spaces of 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)
Neighbourhood 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

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.

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 hybridizing local search methods within the GHH framework is two-fold. 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 hybridization in GHH is similar to that of a memetic algorithm or genetic local search (Krasnogor and Smith, 2005; Krasnogor *et al*, 2006) 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 initialization 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 evaluate the GHH. Firstly, we employ different high-level heuristics (see Section 3) and secondly, we hybridize 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 data sets 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 optimization 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

The different high-level heuristics described in Section 3 are tested on both of the benchmark problems within the unified GHH framework. Owing 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 that are generated by the hybridized method in the next section.

5.1.1. Course timetabling problems Table 5 presents the best and average results by the GHH employing the SDM, ILS, TS and 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 six out of 11 problems, and the best average results on five 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.

5.1.2. 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 six out of 11 problems and the best average results on five out of 11 problems. VNS is the second best algorithm (again on a small scale though) on these problems, obtaining the best results for four out of 11 problems and the best average results for five out of 11 the problems.

5.1.3. Discussion The comparisons between the different high-level heuristics on both course and exam timetabling problems have shown that high-level heuristic moving strategies do not appear to be 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 the 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.

Owing 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.

5.2. Analysis of the hybridizations within the GHH framework

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

Table 5 Different high-level heuristics in the GHH framework on benchmark course timetabling problems (s1–s5: small problems; m1–m5: medium problems)

	*s*1	*s*2	*s*3	*s*4	*s*5	*m*1	*m*2	*m*3	*m*4	*m*5	*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

The best results and the best average results are highlighted and italicized for each problem, respectively. '*x*%': the percentage of runs where no feasible solutions can be obtained.

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)

	*car*91 *I*	*car*92 *I*	*ear*83 *I*	*hec*92 *I*	*kfu*93	*lse*91	*sta83 I*	*tre*92	*ute*92	*uta*93 *I*	*yor*83 *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

5.2.1. 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 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.

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 hybridizing local search upon actual solutions, these soft constraints can be effected on a complete or a partial solution during the solution construction.

5.2.2. Exam timetabling problems Table 8 presents the results on the benchmark exam timetabling problems of the two hybrid GHH approaches. Again, GHH2 outperforms GHH1 over all of the problems on average results, and nine out of 11 problems on the best results. Hybridizing 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.

It can be observed that the best results reported in the literature were obtained by different approaches over the years. Among the eight 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

Table 7 State-of-the-art approaches and GHH hybridizing local search (GHH1: on complete solutions; GHH2: during solution construction) on benchmark course timetabling problems

	*s*1	*s*2	*s*3	*s*4	*s*5	*m*1	*m*2	*m*3	*m*4	*m*5	*Large*
GHH2 best	**0**	**0**	**0**	**0**	**0**	257	259	**192**	235	**112**	0.8/1132
GHH2 avg	*0.2*	*0.6*	*0*	*0.4*	*0.1*	*261*	*273*	*214.5*	*242*	*116*	*1135*
GHH2 time (s)	50	54	48	45	65	19411	15750	18512	18782	9725	20328
GHH1 best	2	2	1	1	0	310	419	332	324	162	0.8/1162
GHH1 avg	2.6	2.8	1	3	2.6	323	428	345	335	182	1162
GHH1 time (s)	155	218	240	171	260	62115	50403	57387	65821	36955	81148
ANT (Socha *et al*, 2002)	1	3	1	1	0	195	184	248	**164.5**	219.5	851.5
CNS (Abdullah *et al*, 2007c)	**0**	**0**	**0**	**0**	**0**	242	**161**	265	181	151	**757**
TS-HH (Burke *et al*, 2003b)	1	2	**0**	1	**0**	**146**	173	267	169	303	1166 80%
RRLS (Socha *et al*, 2002)	8	11	8	7	5	199	202.5	77.5%	177.5	100%	100%

'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 italicized, respectively.

Table 8 State-of-the-art approaches and GHH hybridizing local search (GHH1: on complete solutions; GHH2: during solution construction) on benchmark exam timetabling problems

	*car*91 *I*	*car*92 *I*	*ear*83 *I*	*hec*92 *I*	*kfu*93	*lse*91	*sta*83 *I*	*tre*92	*ute*92	*uta*93 *I*	*yor*83 *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
Abdullah *et al* (2007a)	5.21	4.36	34.87	10.28	**13.46**	10.24	159.2	8.7	26	3.63	**36.2**
Asmuni *et al* (2005)	5.2	4.52	37.02	11.78	15.81	12.09	160.42	8.67	27.78	3.57	40.66
Burke and Newall (2004)	**4.6**	**4.0**	37.05	11.54	13.9	10.82	168.73	8.35	25.83	**3.2**	36.8
Burke *et al* (2004a)	4.8	4.2	35.4	10.8	13.7	10.4	159.1	**8.3**	25.7	3.4	36.7
Caramia *et al* (2008)	6.6	6.0	**29.3**	**9.2**	13.8	**9.6**	158.2	9.4	**24.4**	3.5	36.2
Carter *et al* (1996)	7.1	6.2	36.4	10.8	14.0	10.5	161.5	9.6	25.8	3.5	41.7
Di Gaspero and Schaerf (2001)	6.2	5.2	45.7	12.4	18.0	15.5	160.8	10.0	29.0	4.2	42.0
Merlot *et al* (2002)	5.1	4.3	35.1	10.6	13.5	10.5	**157.3**	8.4	25.1	3.5	37.4

A selection of appropriate methods that includes those that generate the best results among all of the approaches in the literature are highlighted.

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. Hybridizations 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 (ie 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 hybridizing the GHH with local search. Of particular interest is the hybridization 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 hybridization 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 view 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 hybridization with other techniques. Further in-depth performance analysis and understanding may also lead to general observations benefiting general metaheuristic 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 optimization problems. As an iterative constructive approach, the computational time of GHH is usually much longer 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 optimization problems.

References

Asmuni H, Burke EK and Garibaldi J (2005). Fuzzy multiple ordering criteria for examination timetabling. In: Burke EK and Trick M. (eds). *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 3616. Springer: Berlin, pp 334–353.

Abdullah S, Ahmadi S, Burke E and Dror M (2007a). Investigating Ahuja-Orlin's large neighbourhood search for examination timetabling. *OR Spectrum* **29**: 351–372.

Abdullah S, Ahmadi S, Burke EK, Dror M and McCollum B (2007b). A tabu based large neighbourhood search methodology for the capacitated examination timetabling problem. *J Opl Res Soc* **58**: 1494–1502.

Abdullah S, Burke EK and McCollum B (2007c). Using a randomised iterative improvement algorithm with composite neighborhood structures for university course timetabling. In: Doerner KF, Gendreau M, Greistorfer P, Gutjahr WJ, Hartl RF and Reimann M (eds). *Metaheuristics-Progress in Complex Systems Optimization*: Computer Science Interfaces Book Series, Springer Operations Research. Springer: New York, pp 153–172.

Bardadym VA (1996). Computer-aided school and university timetabling: the new wave. In: Burke EK and Ross P (eds). *Practice and Theory of Automated Timetabling*, *Selected Papers from the 1st International Conference*, Springer Lecture Notes in Computer Science, Vol. 1153. Springer: Berlin, pp 22–45.

Bilgin B, Özcan E and Korkmaz EE (2007). An experimental study on hyper-heuristics and exam scheduling. In: Burke EK and Rudová H (eds). *Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 3867. Springer: Berlin, pp 394–412.

Brailsford SC, Potts CN and Smith BM (1999). Constraint satisfaction problems: algorithms and applications. *Eur J Opns Res* **119**: 557–581.

Brucker P and Knust S (2006). *Complex Scheduling*. Springer: Berlin.

Burke EK and Newall J (1999). A multi-stage evolutionary algorithm for the timetabling problem. *IEEE Trans Evol Comput* **3**: 63–74.

Burke EK and Petrovic S (2002). Recent research directions in automated timetabling. *Eur J Opl Res* **140**: 266–280.

Burke EK and Newall J (2004). Solving examination timetabling problems through adaptation of heuristic orderings. *Ann Opns Res* **129**: 107–134.

Burke EK and Kendall G (eds) (2005). *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*. Springer: Berlin.

Burke EK, Jackson K, Kingston JH and Weare R (1997). Automated university timetabling: the state of the art. *Comput J* **40**: 565–571.

Burke EK, Hart E, Kendall G, Newall J, Ross P and Schulenburg S (2003a). Hyperheuristics: an emerging direction in modern search technology. In: Glover F and Kochenberger G (eds). *Handbook of Meta-Heuristics*. Kluwer: Dordrecht, pp 457–474.

Burke EK, Kendall G and Soubeiga E (2003b). A tabu search hyperheuristic for timetabling and rostering. *J Heuristics* **9**: 451–470.

Burke E, Bykov Y, Newall J and Petrovic S (2004a). A time-predefined local search approach to exam timetabling problems. *IIE Trans* **36**: 509–528.

Burke EK, De Causmaecker P, Vanden Berghe G and Van Landeghem H (2004b). The state of the art of nurse rostering. *J Scheduling* (6):441–499.

Burke EK, Kingston J and de Werra D (2004c). Applications to timetabling. In: Gross J and Yellen J (eds). *Handbook of Graph Theory*. Chapman Hall/CRC Press: FL, USA, pp 445–474.

Burke EK, Dror M, Petrovic S and Qu R (2005). Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In: Golden BL, Raghavan S and Wasil EA (eds). *The Next Wave in Computing, Optimization and Decision Technologies*. Springer: Berlin, pp 79–91.

Burke EK, Petrovic S and Qu R (2006). Case based heuristic selection for examination timetabling. *J Scheduling* **9**: 115–132.

Burke EK, McCollum B, Meisels A, Petrovic S and Qu R (2007). A graph-based hyper heuristic for timetabling problems. *Eur J Opl Res* **176**: 177–192.

Burke EK, Eckersley AJ, McCollum B, Petrovic S and Qu R (2008). Hybrid variable neighbourhood approaches to university exam timetabling. *Euro J Opl Res*, forthcoming.

Caramia M, Dell'Olmo P and Italiano GF (2008). Novel local-search-based approaches to university examination timetabling. *INFORMS J Comput* **20**: 86–99.

Carter M and Laporte G (1996). Recent developments in practical exam timetabling. In: Burke EK and Ross P (eds). *Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 1153. Springer: Berlin, pp 3–21.

Carter M and Laporte G. (1998). Recent developments in practical course timetabling. In: Burke EK, and Ross P (eds). *Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 1408. Springer, Berlin, pp. 3–19.

Carter M, Laporte G and Lee S (1996). Examination timetabling: algorithmic strategies and applications. *J Opl Res Soc* **47**: 373–383.

Casey S and Thompson J (2003). GRASPing the examination scheduling problem. In: Burke E, Causmaecker P (eds). *Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 2740. Springer: Berlin, pp 232–246.

Cicirello VA and Smith SF (2004). Heuristic selection for stochastic search optimization: modeling solution quality by extreme value theory. In: Wallace M (ed). *Principles and Practice of Constraint Programming CP 2004*, Lecture Notes in Computer Science, Vol. 3258, Springer: Berlin, pp 197–211.

Crowston WB, Glover F, Thompson GL and Trawick JD (1963). *Probabilistic and parameter learning combinations of local job shop scheduling rules*. ONR Research Memorandum, GSIA, Carnegie Mellon University, Pittsburgh, 117.

Di Gaspero L, Schaerf A (2001). Tabu search techniques for examination timetabling. In: Burke EK and Erben W (eds). *Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 2079. Springer: Berlin, pp 104–117.

Dowsland K, Soubeiga E and Burke EK (2007). A simulated annealing hyperheuristic for determining shipper sizes. *Eur J Opl Res* **179**: 759–774.

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

Eley M (2007). Ant algorithms for the exam timetabling problem. In: Burke EK and Rudova H (eds). *Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 3867. Springer: Berlin, pp 364–382.

Fisher H and Thompson GL (1963). Probabilistic learning combinations of local job-shop scheduling rules. In: Muth JF and Thompson GL (eds). *Industrial Scheduling*. Prentice-Hall: New Jersey, pp 225–251.

Gaw A, Rattadilok P and Kwan RS (2005). Distributed choice function hyperheuristics for timetabling and scheduling. In: Burke EK and Trick M (eds). *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 3616. Springer: Berlin, pp 51–67.

Gomes CP and Selman B (2001). Algorithm portfolios. *Artif Intell* **126**(1–2): 43–62.

Hansen P and Mladenovic N (2001). Variable neighbourhood search: principles and applications. *Eur J Opl Res* **130**: 449–467.

Hansen P and Mladenovic N (2005). Variable neighbourhood search. In: Burke EK and Kendall G (eds). *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techniques*. Springer: Berlin, pp 211–238.

Krasnogor N and Smith JE (2005). A tutorial for competent memetic algorithms: Model taxonomy and design issues. *IEEE Trans Evol Comput* **9**: 474–488.

Krasnogor N, Aragon A and Pacheco J (2006). Memetic algorithms. In: Marti R and Alba E (eds). *Metaheuristics in Neural Networks Learning*. Kluwer: Dordrecht, pp 225–247.

Kwan R (2004). Bus and train driver scheduling. In: Leung J (ed). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press: Boca Raton, FL, USA, Chapter 51.

Lourenco HR, Martin O and Stutzle T (2003). Iterated local search. In: Glover F and Kochenberger G (eds). *Handbook of Metaheuristics*. Kluwer Academic Publishers: Dordrecht, pp 321–353.

McCollum BGC (2007). A perspective on bridging the gap in university timetabling. In: Burke EK and Rudová H (eds). *Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 3867. Springer: Berlin, pp 3–23.

Merlot L, Boland N, Hughes B and Stuckey P (2002). 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, Vol. 2740. Springer: Berlin, pp 207–231.

Meyers C and Orlin JB (2007). Very large-scale neighborhood search techniques. In: Burke EK and Rudova H (eds). *Selected Papers from the 6th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science, Vol. 3867. Springer: Berlin, pp 24–39.

Nareyek A (2003). Choosing search heuristics by non-stationary reinforcement learning. In: Resende MGC and deSousa JP (eds). *Metaheuristics: Computer Decision-Making*. Kluwer Academic Publishers: Dordrecht, pp 523–544.

Nonobe K and Ibaraki T (1998). A tabu search approach to the constraint satisfaction problem as a general problem solver. *Eur J Opl Res* **106**: 599–623.

Petrovic S and Burke EK (2004). University timetabling. In: Leung J (ed). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC: FL, USA, Chapter 45.

Qu R, Burke EK McCollum B, Merlot LTG and Lee SY (2008). A survey of search methodologies and automated system development for examination timetabling. Accepted by *J Scheduling*, DOI: 10.1007/s10951-008-10077-5, accepted for publication.

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

Ross P (2005). Hyper-heuristics. In: Burke EK and Kendall G (eds). *Search Methodologies: Introductory Tutorials in Optimisation and Decision Support Techsiques*. Springer: Berlin, pp 529–556.

Ross P, Marin-Blazquez J and Hart E (2004). Hyper-heuristics applied to class and exam timetabling problems. *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*. IEEE Press, Portland, USA, pp 1691–1698.

Schaerf A (1999). A survey of automated timetabling. *Artif Intell Rev* **13**: 87–127.

Socha K, Knowles J and Sampels M (2002). 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, Vol. 2463. Springer: Berlin, pp 1–13.

Terashima-Marin H, Ross P, Valenzuela-Rendon M (1999). Evolution of constraint satisfaction strategies in examination timetabling. In: Banzhaf W, Daida JM, Eiben AE, Garzon MH, Horava V, Jakiela MJ and Smith RE (eds). *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, Florida USA. Morgan Kaufmann: San Francisco, CA, pp 635–642.

Thompson J and Dowsland K (1998). A robust simulated annealing based examination timetabling system. *Comput Opl Res* **25**: 637–648.

de Werra D (1985). An introduction to timetabling. *Eur J Opl Res* **19**: 151–162.

White GM (2000). Constrained satisfaction, not so constrained satisfaction and the timetabling problem. In: Burke EK and Erben W (eds). *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling*, keynote. Fachhochschule Konstanz, Germany, pp 32–54.

White GM, Xie BS and Zonjic S (2004). Using tabu search with longer term memory and relaxation to create examination timetables. *Eur J Opl Res* **153**: 80–91.

Received July 2007;
accepted July 2008 after two revisions