A Reinforcement Learning: Great-Deluge Hyper-Heuristic for Examination Timetabling

Ender Özcan, University of Nottingham, UK

Mustafa Mısır, Yeditepe University, Turkey

Gabriela Ochoa, University of Nottingham, UK

Edmund K. Burke, University of Nottingham, UK

ABSTRACT

Hyper-heuristics can be identified as methodologies that search the space generated by a finite set of low level heuristics for solving search problems. An iterative hyper-heuristic framework can be thought of as requiring a single candidate solution and multiple perturbation low level heuristics. An initially generated complete solution goes through two successive processes (heuristic selection and move acceptance) until a set of termination criteria is satisfied. A motivating goal of hyper-heuristic research is to create automated techniques that are applicable to a wide range of problems with different characteristics. Some previous studies show that different combinations of heuristic selection and move acceptance as hyper-heuristic components might yield different performances. This study investigates whether learning heuristic selection can improve the performance of a great deluge based hyper-heuristic using an examination timetabling problem as a case study.

Keywords: *Exam Timetabling, Great Deluge, Hyper-Heuristics, Meta-Heuristics, Reinforcement Learning*

1 INTRODUCTION

Meta-heuristics have been widely and successfully applied to many different problems. However, significant development effort is often needed to produce fine tuned techniques for the particular problem or even instance that is under investigation. Hyper-heuristics represent an increasingly popular research direction in search and optimisation (Burke et al., 2003a; Ross, 2005; Chakhlevitch et al., 2008; Özcan et al., 2008; Burke et al. 2009a, 2009b). One of the aims is to at produce more general problem solving techniques, which can potentially be applied to different problems or instances with little development effort. The idea is that a hyper-heuristic approach should able to intelligently choose an appropriate low-level heuristic (from a given repository of heuristics) to be applied at any given time. Thus, in hyper-heuristics, we are interested in adaptively finding solution methods, rather than

DOI: 10.4018/jamc.2010102603

directly producing a solution for whichever search problem we are studying.

Several hyper-heuristics approaches have been proposed in the literature. It is possible to consider methodologies based on *perturbation* low-level heuristics and those based on *construction* low-level heuristics. The latter type builds a solution incrementally, starting with a blank solution and using construction heuristics to gradually build a complete solution. They have been successfully investigated for several combinatorial optimisation problems such as: bin-packing (Tereshima-Marin et al., 2007), timetabling (Terashima-Marin et al., 1999; Burke et al., 2007, Qu et al., 2008), production scheduling (Vazquez-Rodriguez et al., 2007), and cutting stock (Terashima-Marin et al., 2005). On the other hand, approaches based on perturbation heuristics find a reasonable initial solution by some straightforward means (either randomly or using a simple construction heuristic) and then use heuristics, such as shift and swap to perturb solution components with the aim of finding improved solutions. In other words, they start from a complete solution and then search or select among a set of neighbourhoods for better solutions. A class of the most commonly used hyper-heuristics based on perturbation (improvement) low level heuristics is the *choice* hyper-heuristics (heuristic selection methodologies). They have been applied to real world problems, such as, personnel scheduling (Cowling et al., 2001; Burke et al., 2003b), timetabling (Burke et al., 2003b; Dowsland et al., 2007), and vehicle routing problems (Pisinger et al., 2007). In a choice hyper-heuristic framework based on perturbation low level heuristics, search is mostly performed using a single candidate solution. Such hyper-heuristics, iteratively, attempt to improve a given solution throughout two consecutive phases: *heuristic selection* and *move acceptance* as illustrated in Figure 1.

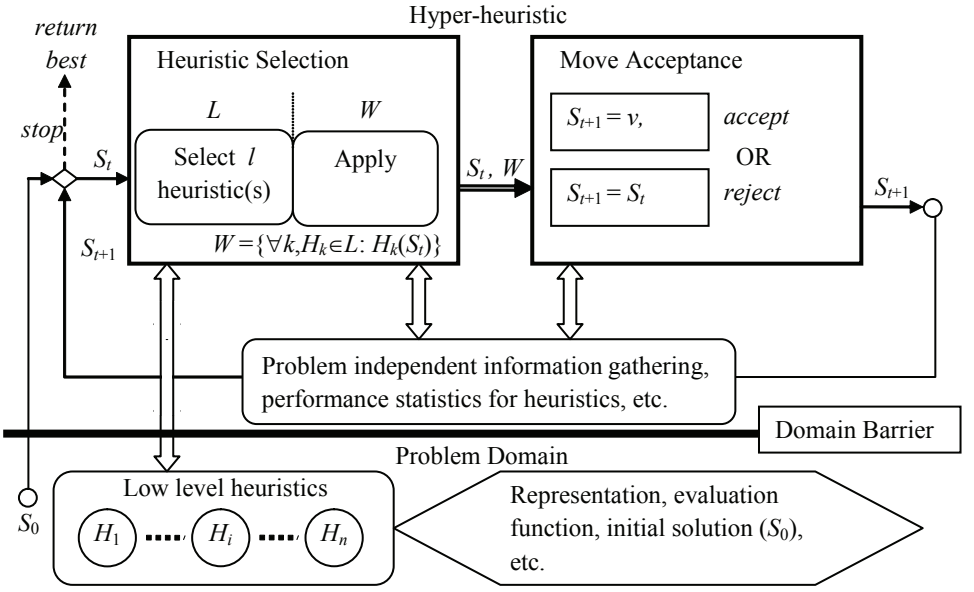
In Figure 1, a candidate solution (S_t) at a given time (t) is modified into a new solution (or solutions) using a chosen heuristic (or heuristics). Then, a move acceptance method is employed to decide whether to accept or reject a resultant solution (R_t). This process is repeated until a predefined stopping condition is met. Only problem independent information flow is allowed between the problem domain and hyper-heuristic layers. Unless, we specifically say otherwise, a choice hyper-heuristic refers to a hyper-heuristic that operates on a set of perturbation low level heuristics from this point onwards. Moreover, such a hyper-heuristic will be denoted as *heuristic selection − move acceptance* based on its components.

Great deluge is a well-known acceptance strategy (Dueck, 1993; Burke et al., 2003). Bilgin et al. (2007) reported that hyper-heuristics formed by different combinations of heuristic selection and move acceptance methods might yield different performances. Moreover, simple random−great deluge delivered a similar performance to the best approach; namely, choice function – simulated annealing for examination timetabling. Obviously, simple random receives no feedback at all during the search to improve upon the heuristic selection process. Hence, in this study, g*reat-deluge* is preferred as the move acceptance component within a choice hyper-heuristic framework to investigate the effect of learning heuristic selection on its overall performance for solving the same examination timetabling problem as formulated in Bilgin et al. (2007). The learning mechanisms, inspired by the work by Nareyek (2003), are based on weight adaptation.

2 Hyper-Heuristics and Learning

Although hyper-heuristic as a term has been introduced recently (Denzinger et al., 1997), the origins of the idea date back to the early 1960s (Fisher et al., 1961). A hyper-heuristic operates at a high level by managing or generating low level heuristics which operate on the problem domain. Meta-heuristics have been commonly used as hyper-heuristics. A hyper-heuristic can conduct a single point or multi-point search. Population based meta-heuristics which perform multi-point search, such as learning classifier systems (Marín-Blázquez and Schulenburg, 2005), evolutionary algorithms (Cowling et al.,

Figure 1. A hyper-heuristic framework based on a single point search, where S_t denotes a candidate solution at time t, H_i is the i^{th} low level heuristic, R_t is the resultant solution after applying a set of selected low level heuristics that goes into the move acceptance process



2002; Han et al., 2003; Pillay and Banzhaf, 2008), genetic programming (Keller et al., 2007; Burke et al., 2009a), ant colony optimisation (Cuesta-Canada et al., 2005; Chen et al., 2007) have been applied to a variety of combinatorial optimisation problems as hyper-heuristics. Distributed computing methods can also be used to perform multi-point search (Rattadilok et al., 2004; Rattadilok et al., 2005; Ouelhadj et al., 2008). Özcan et al. (2008) presented different hyper-heuristic frameworks showing that a matching performance to memetic algorithms can be achieved. In this study, the choice hyper-heuristic framework as presented in Figure 1 is studied. The primary components of such hyper-heuristics are heuristic selection and move acceptance.

A major motivating feature of hyper-heuristic research is the aim to facilitate applicability to different problem instances having different characteristics as well as different problem domains. With this goal in mind, machine learning techniques are vital for hyper-heuristics to make the right choices during the heuristic selection process. Learning can be achieved in an *offline* or *online* manner. An offline learning hyper-heuristic requires training over a set of problems, before it is used to solve the unseen problem instances. For example, Burke et al. (2006) use a case based reasoning system as a hyper-heuristic for solving course and examination timetabling problems. An online learning hyper-heuristic learns through the feedback obtained during the search process while solving a given problem. Most of the existing online learning hyper-heuristics incorporate *reinforcement learning* (Kaelbling et al., 1996; Sutton et al., 1998). A reinforcement learning system interacts with the environment and changes its state via a selected action in such a way as to increase some notion of long term reward. Hence, a learning hyper-heuristic maintains a utility value obtained through predetermined reward and punishment schemes for each low level heuristic. A heuristic is selected based on the utility values of the low level heuristics in

hand at each step. Remembering and forgetting represent core ingredients of learning. Remembering can be achieved through reward and punishment schemes. Forgetting can be achieved through the use of lower and upper bounds on the utility values. Some reinforcement learning methods use weighted average of the learnt utility values. A dynamic weighting scheme can be employed which favours the outcome of the most recent actions or choices. Reward and punishment schemes are allowed to use different adaptation rates in the case of an improving and worsening move, respectively. For example, the utility value of a selected heuristic can be increased at a constant rate linearly whenever there is an improvement after it is employed, otherwise the utility value can be decreased at a different rate, or it can even be kept constant. Initialisation of the utility values, lower and upper bounds for them along with a memory adjustment scheme (weighting) represent the remainder of the constituents for a reinforcement learning based hyper-heuristic.

Some previously studied heuristic selection methods are summarised in Table 1. Simple random, random gradient, random permutation gradient, greedy and choice function heuristic selection methods are presented in Cowling et al. (2001a). All these approaches can be considered to be learning heuristic selection methods, except simple random. In Cowling et al. (2001b), a parameter-free choice function was presented. As a problem domain, sales summit scheduling was used in both studies. Cowling and Chakhlevitch (2003) investigated peckish heuristic selection strategies that eliminated the selection and application of all low level heuristics as in greedy heuristic selection.

Nareyek (2003) investigated reinforcement learning using different reward/penalty schemes and heuristic selection strategies on the Orc Quest problem and in the logistics domain. Additive/subtractive adaptation rates combined with heuristic selection using the maximal utility generated better results as opposed to a fair random choice (softmax, roulette wheel). All heuristics were assigned to a utility value of 0 initially and raw utility values were maintained. Upper and lower bounds were defined for the utility values. In Burke et al. (2003b), reinforcement learning was combined with tabu search in a hyper-heuristic and applied to the personnel rostering and timetabling problems. The aim of this modification was to prevent the selection of some heuristics for a while by inserting them into a variable-length tabu list. A non-tabu heuristic with the highest utility value was chosen at each step.

Some studies concentrate on move acceptance in hyper-heuristics rather than upon heuristic selection methods, as accepting a move turns out to be an extremely important decision. In Cowling et al. (2001), heuristic selection methods are combined with either all moves accepted or with only an improving moves accepted strategy. On the other hand, Ayob and Kendall (2003) proposed three different Monte Carlo move acceptance strategies based on the objective value change due to the move, time (units), number of consecutive non-improving moves. Simple random was used as a heuristic selection within the hyper-heuristic for solving the component placement problem. The best move acceptance turned out to be *exponential Monte Carlo with counter*. One of the well known move acceptance strategies is *simulated annealing* (SA) (Kirkpatrick, 1983). The improving moves or the moves that generate an equal quality solution are accepted, while a worsening move is not rejected immediately. Acceptance of a given candidate solution is based on a probabilistic framework that depends on the objective value change and a temperature that decreases in time (cooling). The difference between exponential Monte Carlo with counter and the simulated annealing is that the latter one uses this *cooling schedule* while the former does not. Bai and Kendall (2003) investigated the performance of a simple random – simulated annealing hyper-heuristic on a shelf space allocation problem. Anagnostopoulos et al. (2006) applied a similar hyper-heuristic to a set of travelling tournament problem instances embedding a reheating scheme into the simulated annealing move acceptance. In Bai et al. (2007a), a reinforcement learning scheme is

Table 1. Description of a set of heuristic selection methods used within choice hyper-heuristics

Name	Description
Simple Random	Choose a low level heuristic randomly
Random Descent	Choose a low level heuristic randomly and employ the same heuristic as long as the candidate solution in hand is improved
Random Permutation Descent	Generate a random permutation of low level heuristics and form a cyclic list. Starting from the first heuristic, employ it repeatedly until a worsening move is hit, then go to the next heuristic in the list.
Greedy	Apply all low level heuristics to the same candidate solution separately and choose the heuristic that generates the best change in the objective value
Peckish	Apply a subset of all low level heuristics to the same candidate solution and choose the heuristic that generates the best change in the objective value
Choice Function	Dynamically score each heuristic weighing their individual performance, combined performance with the previously invoked heuristic and time passed since the last call to the heuristic at a given step. Then, a heuristic is chosen based on these scores.
Reinforcement Learning	Each heuristic carries a utility value and heuristic selection is performed based on these values. This value gets updated at each step based on the success of the chosen heuristic. An improving move is rewarded, while a worsening move is punished using a preselected adaptation rate.
Tabu Search	This method employs the same strategy as Reinforcement Learning and uses a tabu list to keep track of the heuristics causing worsening moves. A heuristic is selected which is not in the tabu list.

combined with simulated annealing with reheating as a hyper-heuristic and applied to three different problem domains: nurse rostering, course timetabling and 1D bin packing.

In (Dueck, 1993), two move acceptance strategies, namely *great deluge* (GD) and *record-to-record travel* that accept worsening moves based on a dynamic threshold value were presented. Kendall and Mohamad (2004) utilised a simple random – great deluge hyper-heuristic to solve a mobile telecommunication network problem. Great deluge uses a threshold that decreases in time at a given rate (e.g., linearly) to determine an acceptance range for the solution qualities based on three main parameters: (i) the maximum number of iterations (or total time), (ii) the number of iterations (or time) passed, and (iii) an expected range for the maximum fitness change between the initial and final objective value (e.g., lower bound). In the case of an improving move, it is accepted, while a worsening move is accepted only if the objective value of the resultant candidate solution is less than the computed threshold at a given iteration. Kendall and Mohamad (2004) used an iteration based threshold formula with a maximum number of iterations as a termination criterion aiming a quadratic running time for the overall algorithm.

Bilgin et al. (2007) employed different heuristic selection and move acceptance mechanisms and used their combinations as hyper-heuristics. The results showed that a simple random – great deluge hyper-heuristic was the second best after choice function – simulated annealing, considering the average performance of all hyper-heuristics over a set of examination timetabling problems. Consequently, a hyper-heuristic without learning delivered a comparable performance to another one with a learning mechanism. Therefore, in this study, reinforcement learning is combined with great deluge to observe the effect of learning heuristic selection on the overall performance of the

hyper-heuristic for solving the same problem. All the runs during the experiments in (Bilgin et al., 2007) were restricted to 600 seconds; hence, the threshold is computed based on the CPU time within the great deluge move acceptance strategy. If a heuristic takes less time, then the threshold value will be lower as compared to the one that takes longer time. This hyper-heuristic differs from the one that Kendall and Hussin (2005) have investigated, as their hyper-heuristic embeds a tabu list approach to keep the chosen heuristic from getting selected again for a number of steps (tabu duration) into reinforcement learning as a heuristic selection. Moreover, the low level heuristics contained a mixture of thirteen different construction and perturbation low level heuristics.

Özcan et al. (2009) combined different heuristic selection methods with a late acceptance strategy, a new method that is initially presented as a local search for solving examination timetabling problems. Late acceptance requires a single parameter and it is a memory based approach. A trial solution is compared with a previously visited solution at a fixed distance apart from the current step in contrast to the conventional approaches that usually compare the trial solution with a current one. The trial solution is accepted, if there is an improvement over this previously visited solution. The results showed that reinforcement learning, reinforcement learning with tabu list or choice function heuristic selection methods did not improve the performance of the hyper-heuristic if late acceptance is used. Choosing a heuristic randomly at each step performed the best. More on hyper-heuristics can be found in Burke et al. (2003a), Ross (2005), Özcan et al. (2008), Chakhlevitch and Cowling (2008), Burke et al. (2009a, 2009b, 2009c).

3 THE EXAMINATION TIMETABLING PROBLEM

Examination timetabling is a challenging real world problem addressed by educational institutions. The goal is to find the best assignment of available timeslots and possibly other resources, such as rooms for each examination subject to a range of constraints. There are two types of constraints: hard and soft constraints. Hard constraints must not be violated under any circumstances and a solution which satisfies them is called a feasible solution. For example, a student cannot take any pair of his/her examinations at the same time. Soft constraints reflect preferences and their violation is allowed, but the goal is to minimise it. For example, a number of timeslots might be preferred in between the examinations of a student scheduled to the same day.

3.1 Previous Work

Researchers have been studying various aspects of examination timetabling problems since the early 1960s (Cole, 1964; Broder, 1964). Examination timetabling problems are NP-complete (Even, 1976). Since the search space of candidate solutions grows exponentially with respect to the number examinations to be scheduled, many different non-traditional approaches (e.g., meta-heuristics) have been investigated for solving a variety of examination timetabling problems. Tables 2 and 3 provide some illustrative examples of these approaches.

Many examination timetabling problems are studied from a practical point of view, as they arise due to practical needs within institutions. It is worth pointing out that different institutions have very different requirements (Burke et al., 1996a). One consequence of this is that there is a variety of examination timetabling problems in the literature (Table 3; see Qu et al., 2009). Carter et al. (1996b) introduced one of the widely used examination timetabling data sets which was originally made up of 13 real world problems. Özcan et al. (2005) introduced an examination timetabling problem at Yeditepe University. In this initial study, different memetic algorithms were described. A type of violation directed hill climbing (Alkan and Özcan, 2003) was also investigated as a part of the memetic algorithm which turned out to be the best choice. A survey on examination timetabling is provided by

Table 2. Different approaches to examination timetabling

Approach	Representative Reference(s)
Decomposition and/or construction heuristics	Qu and Burke (2007);
Simulated annealing	Thompson and Dowsland (1998); Merlot et al. (2002);
Genetic algorithms and constraint satisfaction	Marin (1998)
Grouping genetic algorithm	Erben (2001)
Iterative greedy algorithm	Caramia et al. (2001)
Tabu search	Di Gaspero and Schaerf (2001); Burke et al. (2005)
Multiobjective evolutionary algorithm	Paquete and Fonseca (2001); Cheong et al. (2007)
Greedy randomised adaptive search procedure	Casey and Thompson (2003)
Adaptive heuristic ordering strategies	Burke and Newall, (2004)
Very large neighbourhood search	Abdullah et al. (2007)
Fuzzy reasoning	Petrovic et al. (2005)
Variable neighbourhood search	Qu and Burke (2005)
Ant colony optimisation	Dowsland and Thompson (2005)
Hybrid heuristics	Azimi (2005), Ersoy et al. (2007)
Neural network	Corr et al. (2006)
Case based reasoning based investigations	Petrovic et al. (2007)
Alternating stochastic-deterministic local search	Caramia and Dell'Olmo (2007)
Hyper-heuristics	Burke et al. (2006), Pillay and Banzhaf (2008)

Table 3. Some examination timetabling problems from different universities and the initial approaches proposed to solve them

Institution	Reference	Approach
University of Nottingham	Burke et al. (1995)	Memetic algorithm
Middle East Technical University	Ergul (1996)	Genetic algorithm
École de Technologie Supérieure	Wong et al. (2002)	Genetic algorithm
University of Melbourne	Merlot et al. (2002)	A multi-phase hybrid algorithm
University of Technology MARA	Kendall and Hussin (2005)	Hyper-heuristic
Yeditepe University	Özcan et al. (2005)	Memetic algorithm

Qu et al. (2009). Carter (1986) and Carter et al. (1996a, 1996b) provide earlier surveys on examination timetabling.

3.2 Problem Description

In this study, we deal with the examination timetabling problem at Yeditepe University. This specific problem requires finding the best timeslots for a given set of examinations under four hard constraints and a soft constraint. The hard constraints are as follows:

- **Scheduled examination restriction:** Each examination must be assigned to a timeslot only once.

- **Unscheduled examination restriction:** All the examinations must be scheduled.
- **Examination clash restriction (C_1):** A student cannot enter into more than one examination at a given time.
- **Seating capacity restriction (C_2):** The number of students seated for all exams at a timeslot cannot be more than a given capacity.

The soft constraint is as follows:

- **Examination spread preference (C_3):** A student should have at least a single timeslot in between his/her examinations in the same day.

Let E represent the set of examinations $E=\{e_1,\ldots,e_j,\ldots,e_n\}$ and let S denotes the ordered list of timeslots to be assigned to the examinations, $S=\{t_1,\ldots, t_k,\ldots,t_p\}$. An array $A=\{a_1,\ldots, a_j,\ldots,a_n\}$ is used as a direct representation of a candidate solution, where each entry $a_j=t_k$, $t_k \in S$, indicates that e_j is assigned to a timeslot t_k in S. Hence, scheduled and unscheduled examination restrictions are resolved by using this direct and complete representation that encodes a timeslot for each given examination. The quality of a given timetable (TT) with respect to a set of students and the courses upon which they enrolled (SR) is determined by calculating the weighted average of constraint violations.

$$quality(TT) = \frac{-1}{1+\sum_{\forall i} violations(C_i, TT, SR)\ w_i} \quad (1)$$

where $i=\{1,2,3\}$ and *violations* measures the violations due to a constraint C_i in TT considering SR.

The performances of a set of Reinforcement Learning – Great Deluge hyper-heuristics are investigated over the Yeditepe University and Toronto benchmarks (Carter et al., 1996b). Yeditepe University (Faculty of Engineering) data set contains real problem instances from each semester in three consecutive years. Bilgin et al. (2007) modified the initial data set provided in Özcan et al. (2005) with new properties and also generated a variant of Toronto benchmarks that fits into the problem formulation. The Yeditepe University data sets and Toronto benchmarks can be obtained from http://www.cs.nott.ac.uk/~exo/research/TTML/ and http://www.cs.nott.ac.uk/~rxq/data.htm, respectively. The number of exams determines the size of the search space to be explored, but the difficulty of a given problem might change with respect to some other characteristics, such as the number of students or conflict density (ratio of the number of examination pairs that should not clash to the total number of examination pairs) that might implicitly or explicitly restrict the search space containing feasible solutions. Such properties for each experimental data are provided in Table 4.

4 THE REINFORCEMENT LEARNING – GREAT DELUGE HYPER-HEURISTIC

Reinforcement Learning (RL) is a general term for a set of widely used approaches that provide a way to learn how to behave when an action comes or "*how to map situations to actions*" (Sutton and Barto, 1998) through *trail-and-error* interactions (Kaelbling et al., 1996). A choice hyper-heuristic combining reinforcement learning heuristic selection and great deluge move acceptance is implemented as shown in Figure 2. As suggested in Nareyek (2003), additive adaptation rate that increments the utility value of the low level heuristic is used in the case of an improvement as a reward at step 14. This value is tested against three different negative adaptation rates, namely subtractive, divisional and root, denoted as r_1, r_2 and r_3, respectively for the punishment of a heuristic causing a worsening move at step 17, where u_i is the utility value of the ith low level heuristic:

$$r_1 : u_i = u_i - 1 \quad (2)$$

Table 4. Properties of Yeditepe and modified Toronto benchmark problem instances

Data Set	Instance	Exams	Students	Enrolment	Conflict Density	Days	Capacity
Yeditepe	yue20011	126	559	3486	0.18	6	450
	yue20012	141	591	3708	0.18	6	450
	yue20013	26	234	447	0.25	2	150
	yue20021	162	826	5755	0.18	7	550
	yue20022	182	869	5687	0.17	7	550
	yue20023	38	420	790	0.20	2	150
	yue20031	174	1125	6714	0.15	6	550
	yue20032	210	1185	6833	0.14	6	550
Toronto	car91 I	682	16925	56877	0.13	17	1550
	car92 I	543	18419	55522	0.14	12	2000
	ear83 I	190	1125	8109	0.27	8	350
	hecs92 I	81	2823	10632	0.42	6	650
	kfu93	461	5349	25118	0.06	7	1955
	lse91	381	2726	10918	0.06	6	635
	pur93 I	2419	30029	120681	0.03	10	5000
	rye92	486	11483	45051	0.07	8	2055
	sta83 I	139	611	5751	0.14	4	3024
	tre92	261	4360	14901	0.18	10	655
	uta92 I	622	21266	58979	0.13	12	2800
	ute92	184	2749	11793	0.08	3	1240
	yor83 I	181	941	6034	0.29	7	300

$$r_2 : u_i = u_i / 2 \quad (3)$$

$$r_3 : u_i = \sqrt{u_i} \quad (4)$$

Memory length is implemented not only in terms of adaptation rates, but also by using a lower and an upper bound on the utility values. We experimented with four different ranges in $[0,number\_of\_heuristics\times(5i)]$, $i=\{1,2,3,4\}$. It is assumed that these bounds are checked during steps 14 and 17. Optimistic initial utility values are utilised and all utilities are set to $\lfloor 0.75\times upper\ bound \rfloor$ at step 3 to support exploration. As the environment might change dynamically, bounds on the utility values are essential in order to encourage exploration in further steps. Reinforcement learning is based on the idea that heuristics obtaining large rewards should be more likely to be selected again, while heuristics getting small rewards should be less likely to be selected again. The reinforcement scheme used returns the same reward for all heuristic choices and we use the maximal utility value to select a heuristic. Note that selecting the heuristic with this strategy (denoted as *max*) is reported in (Nareyek, 2003) to be the best choice for step 9. If there are multiple low level heuristics under consideration, since their utility values are the same, then a random choice is made. Another approach to decide whether a given total reward is small or large can be achieved by comparing that value to a relative *reference reward*, such as the average of all utility values. In addition to the maximal

Figure 2. Pseudocode of the Reinforcement Learning – Great Deluge hyper-heuristic

```
RL–GD ALGORITHM
Input – n: number of heuristics, u: array holding utility value for each heuristic, totalTime
1.  // initialisation
2.  Generate a random complete solution S_current ;
3.  Initialise utility values;
4.  f_current = f_0 = quality( S_current );
5.  startTime = t = time(); level = f_current
6.  // main loop executes until total running time allowed is exceeded
7.  while ( t < totalTime ) {
8.      // heuristic selection
9.      i = selectHeuristic( u ); // select a heuristic using the utility values
10.     S_temp = applyHeuristic( i );
11.     f_temp = quality( S_temp );
12.     t = time() – startTime;
13.     // move acceptance
14.     if (f_temp < f_current ) then {
15.             u_i = reward( u_i ); // improving move
16.             S_current = S_temp;
17.     } else {
18.             u_i = punish( u_i ); // worsening move
19.             if ( f_temp < qualityLB + (f_0 – qualityLB)(1 – t/totalTime) ) then
20.                     S_current = S_temp; // accept the move else reject the move
21.     }
22. }
```

utility, another heuristic selection scheme that chooses a low level heuristic randomly from the ones that are over (and equal to) the average, denoted as *overAvr* is implemented. The lower bound (*qualityLB*) is set to -1 at step 19 considering the evaluation function (Equation 1) during the experiments. The reinforcement learning heuristic selection methods using the negative adaptation rates r_1, r_2 and r_3 are referred to as RL_1, RL_2 and RL_3, respectively.

In this study, we employed four low level heuristics (Bilgin et al., 2007). Three of them H_1, H_2 and H_3 are associated with three constraints C_1, C_2 and C_3, respectively. They probe constraint based neighbourhoods using tournament selection to resolve violations of a corresponding constraint only. Each low level heuristic operates as follows:

1. $\boldsymbol{H_1(H(x))}$**:** This heuristic chooses a number of examinations randomly that violate x=C_1 and this number is referred to as *toursize*1. Then, the examination causing the largest number of violations is selected. This examination is reassigned to a timeslot from a randomly selected set of timeslots (*toursize*2) which generates the least number of x=C_1 violations.
2. $\boldsymbol{H_2}$**:** Using a tournament strategy, a number of timeslots (*toursize*3) with capacity constraint violations are selected. Examinations in the timeslot that has the largest number of violations are marked for further processing. The examination with the largest number of enrolled students is rescheduled. Then, this examination is reassigned to a timeslot from a randomly selected set of timeslots (*toursize*4) which generates the least amount of C_2 violations.
3. $\boldsymbol{H_3}$**:** This heuristic employs the same strategy as described in $H(x)$ with x=C_3.
4. $\boldsymbol{H_4}$**:** This heuristic makes a pass over all the examinations and reschedules the examination under consideration with a probability of 1/*number_of_examinations*.

5 EXPERIMENTAL RESULTS

The experiments were performed on a Pentium IV 3 GHz LINUX (Fedora Core 8) PC with 2 GB memory. Each hyper-heuristic is tested on each instance for 50 trials and each trial is terminated after 600 CPU seconds. Initial experiments were performed for parameter tuning. Unless mentioned otherwise, the utility value upper bound is fixed as 40 and *max* is used as the utility based heuristic selection

strategy within the reinforcement learning hyper-heuristics. A sample run is performed for sta83 I using a reinforcement learning – great deluge hyper-heuristic. Figure 3 illustrates the change in utility for each low level heuristic and the improvement based on different negative adaptation rates for this run. If a low level heuristic worsens the solution after a number of successive improving moves, the best heuristic still gets a chance to operate on the candidate solution. The frequency of that chance is determined by the negative adaptation rate. For example, H_3 gets selected more frequently when the adaptation rate is subtractive(/divisional) as compared to divisional(/root) before the optimistic utility values of all heuristics reduces toward the lower bound (see Figure 3). The more severe (high) this rate is, the more exploration of different heuristics is favoured. All the low level heuristics get invoked within tens of steps while using divisional and root adaptation rates (Figure 3. (b) and (c)), whereas only two heuristics get invoked while using a subtractive adaptation rate (Figure 3. (a)).

The results show that all low level heuristics are valuable in improving a candidate solution. It seems that the quality of a solution is improved slowly whenever a slow negative adaptation rate is used. Naturally, there is still the chance of getting stuck at a local optimum in the long run. A low level heuristic at any step is chosen with a probability in {1.00, 0.50, 0.33, 0.25}. The reinforcement learning heuristic selection arranges these probabilities dynamically during the search process. As the search stagnates and a local optimum is found, the probability of 0.25 is used more frequently while selecting a low level heuristic.

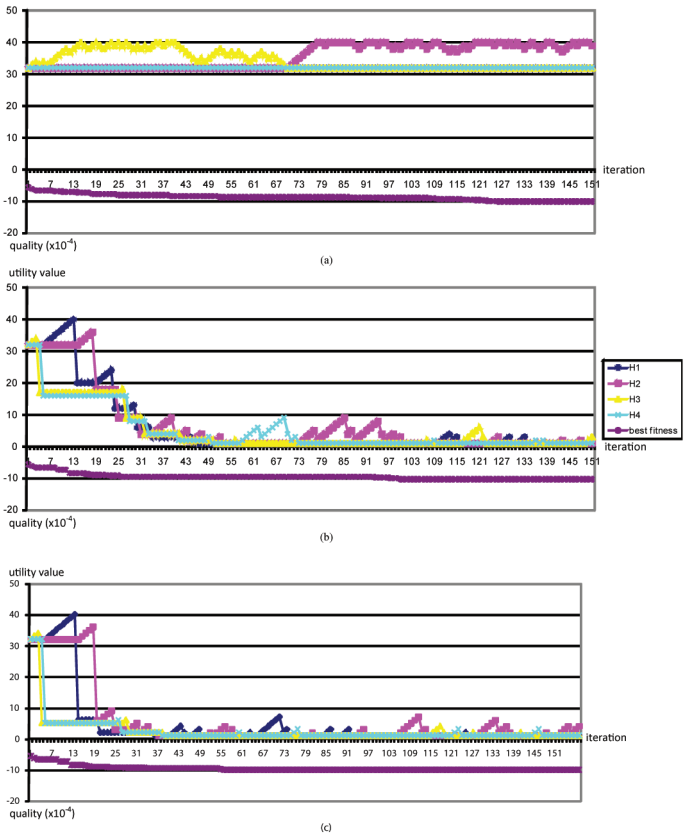
In order to observe the effect of memory length via different combinations of negative adaptation {subtractive, divisional, root} and upper bound for the utility values {20, 40, 60, 80}experiments have been performed on modified Toronto problem instances. As a total twelve different choices are executed for each data and each choice is ranked from 1 (best) to 12 (worst) using the results from the runs. The average rank of a choice over all data and the related standard deviation are provided in Figure 4.

Determining the best adaptation rate, which is also vital to adjust the memory length, seems to be a key issue in fully utilising a reinforcement learning scheme within a hyper-heuristic. Different adaptation rates might yield different performances. The results show that the RL_1 heuristic selection method with a utility upper bound of 40 delivers the best average performance when combined with the great deluge method as a hyper-heuristic. Yet, this performance variation is not statistically better than the rest.

Using the best configuration from the previous set of experiments, another experiment is performed over modified Toronto problem instances to compare the average performances of utility based heuristic selection schemes; *max* and *overAvr*. Figure 5 summarises the experimental results. Maximal utility selection performs slightly better than *overAvr* with an average rank of 1.42 for the modified problem instances {car91 I, car92 I, kfu93, lse91, pur93 I, rye92, ute92}. There is a tie for sta83 I. Still, the performance difference between *max* and *overAvr* is not statistically significant. There might still be potential for a future use of this approach, as, in general, *overAvr* shows success in solving problem instances with relatively high conflict densities.

The reinforcement learning heuristic selection method, which is referred to as RL_1, utilises additive reward and subtractive punishment schemes with a utility upper bound of 40 and *max*. RL_1 is combined with great deluge and tested against simple random – great deluge hyper-heuristic during the final set of experiments. The results are provided in Table 5. The percentage improvement in the table uses whichever approach generates a better result (average best of fifty trials) as the baseline for comparison. The simple random – great deluge hyper-heuristic generates better average performance for eight problem instances. It is especially successful in solving the Yeditepe problem instances which are smaller and have low conflict densities when

Figure 3. Plot of utility value for each low level heuristic and quality versus iteration on sta83 I using the reinforcement learning – great deluge hyper-heuristic based on (a) subtractive, (b) divisional and (c) root negative adaptation rates with max, utility upper bound=40, respectively



compared with the modified Toronto instances. Yet, RL_1 improves the performance of simple random hyper-heuristic with the great deluge move acceptance on eleven problem instances (out of twenty one problem instances), and for two problem instances there is a tie.

Finally, RL_1 – great deluge is compared to two previous studies. Bilgin et al. (2007) showed that the choice function – simulated annealing hyper-heuristic, out of thirty five approaches, performs the best for examination timetabling. In a recent study, Özcan et al. (2009) introduced a new move acceptance strategy that can be used in hyper-heuristics. The experiments resulted in the success of a simple random – late acceptance hyper-heuristic, performing even better than choice function – simulated annealing. Both of these approaches are compared to the RL_1 – great deluge in Table 6.

A hyper-heuristic learns how to make *good* moves through both heuristic selection and move acceptance. If a move is rejected, then the selected heuristic is annulled. Hence, if a hyper-heuristic uses a simple random heuristic selection, it does not imply that there is no learning within that hyper-heuristic. The late acceptance strategy uses a fixed length memory to hold the quality of some previously visited solutions. Simple random heuristic selection diversifies the search, while the late acceptance

Figure 4. Average rank of hyper-heuristics using different heuristic selection methods {RL_1, RL_2, RL_3} with a utility upper bound of (a) 20, (b) 40, (c) 60 and (d) 80 over modified Toronto instances

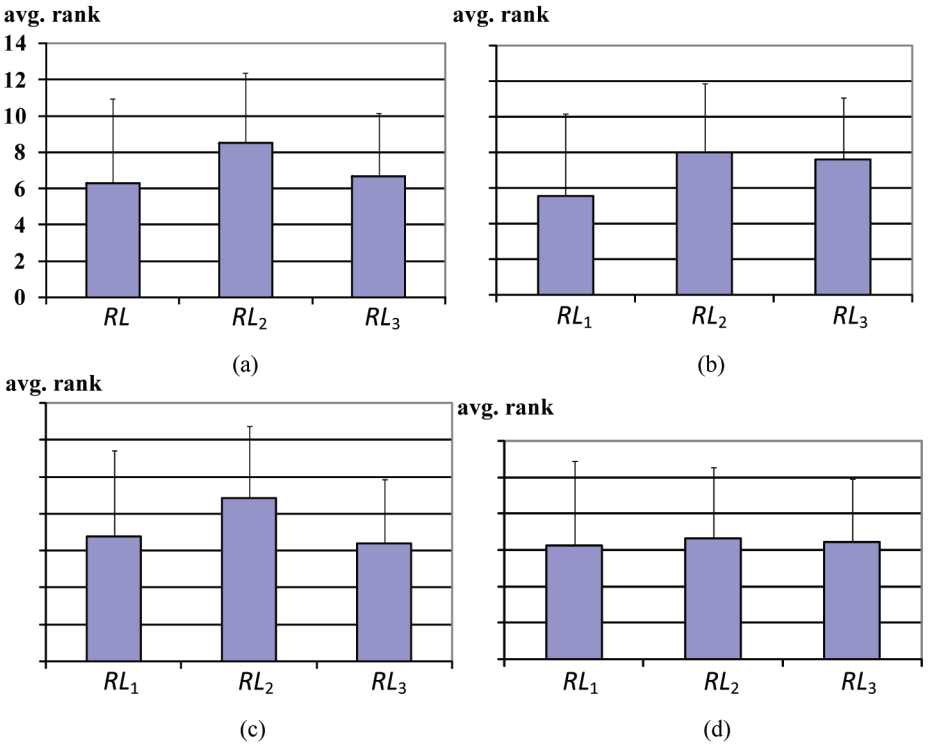


Figure 5. Comparison of utility value based heuristic selection schemes over modified Toronto instances based on their average ranks

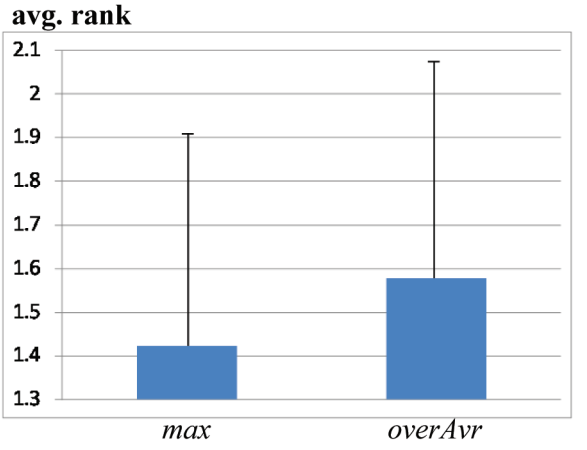


Table 5. Comparison of reinforcement learning and simple random heuristic selection within a hyper-heuristic using the great deluge acceptance move method. "≥" and "≈" indicate "is better than" and "delivers a similar performance", respectively. 'Percentage improvement' uses the average best quality obtained in fifty runs for the better approach as the baseline for comparison.

Instance	Comparison	%-improv.
car91 I	RL_1–GD ≥ SR–GD	1.70
car92 I	RL_1–GD ≥ SR–GD	1.68
ear83 I	RL_1–GD ≥ SR–GD	2.02
hecs92 I	**SR–GD ≥ RL_1–GD**	**11.85**
kfu93	RL_1–GD ≥ SR–GD	2.09
lse91	RL_1–GD ≥ SR–GD	2.47
pur93 I	**SR–GD ≥ RL_1–GD**	**0.32**
rye92	RL_1–GD ≥ SR–GD	3.43
sta83 I	RL_1–GD ≥ SR–GD	0.06
tre92	**SR–GD ≥ RL_1–GD**	**5.05**
uta92 I	**SR–GD ≥ RL_1–GD**	**0.23**
ute92	RL_1–GD ≥ SR–GD	0.28
yor83 I	RL_1–GD ≥ SR–GD	1.13
yue20011	RL_1–GD ≈ SR–GD	0.00
yue20012	RL_1–GD ≥ SR–GD	0.53
yue20013	RL_1–GD ≈ SR–GD	0.00
yue20021	**SR–GD ≥ RL_1–GD**	**1.49**
yue20022	**SR–GD ≥ RL_1–GD**	**0.83**
yue20023	**SR–GD ≥ RL_1–GD**	**0.71**
yue20031	RL_1–GD ≥ SR–GD	5.26
yue20032	**SR–GD ≥ RL_1–GD**	**0.88**

strategy intensifies the search process by approving the better moves based on its memory. This course of action can also be considered to be learning. Reinforcement learning not only improves on simple random heuristic selection when combined with great deluge but it also generates better results as compared to another learning hyper-heuristic; choice function – simulated annealing. Moreover, its performance is comparable to the performance of the simple random – late acceptance hyper-heuristic.

6 CONCLUSION

In this article, a hyper-heuristic can be thought of as a methodology that guides the search process at a high level by controlling a set of perturbation low level heuristics. A single point search hyper-heuristic framework that combines successive stages of heuristic selection and move acceptance is employed during the experiments. It has been observed in Bilgin et al. (2007) that simple random heuristic selection in a great deluge based hyper-heuristic performed well for solving an examination timetabling

Table 6. Comparison of RL_1 – great deluge to the previous studies; (1) choice function – simulated annealing (Bilgin et al., 2007), (2) simple random – late acceptance (Özcan et al., 2009). Each approach is ranked from 1 (best) to 3 (worst) for each modified Toronto problem instance.

Instance	RL_1–GD	(1) CF–SA	(2) SR–LAS
car91 I	2	3	1
car92 I	1	3	2
ear83 I	1	2	3
hecs92 I	1	3	2
kfu93	1	3	2
lse91	2	3	1
pur93 I	2	3	1
rye92	2	3	1
sta83 I	1	3	2
tre92	1	3	2
uta92 I	2	3	1
ute92	3	1	2
yor83 I	2	3	1
avg.	1.62	2.77	1.62

problem encountered at Yeditepe University every semester. The same problem is used as a case study to investigate reinforcement learning with different components as heuristic selection methods in place of simple random.

There are different ways of implementing the great deluge move acceptance. In this study, a linear decreasing rate is adopted as suggested in previous studies (Bilgin et al., 2007; Kendal and Mohamad, 2004). Additionally, CPU time is used as a termination criterion and hence, CPU time is used as a part of the *level* decreasing scheme. This level sets a goal for the chosen heuristic in the case of a worsening move after its invocation. The resultant *bad* move is required to have a better quality than the goal, otherwise it is rejected. Using CPU time within the great deluge provides an additional side benefit. The running times of low level heuristics might be different. If a heuristic takes a short time to execute, then the expectation on the quality of the resultant move is lower as compared to a heuristic which takes a longer time to execute. This strategy seems to be viable and great deluge based on this strategy performs well as a hyper-heuristic component (Bilgin et al., 2007; Özcan et al., 2008).

A learning hyper-heuristic usually attempts to follow the best moves within a given period of time using some type of memory to make better future decisions. Bai et al. (2007b) observed that memory length is vital in learning and that the use of a learning mechanism with *short* term memory combined with a simulated annealing move acceptance generated the best results in their experiments over a set of course timetabling problems. They used weighted adaptation and tested various *learning rates* that adjust the influence of rewards compiled at different stages of search. In this study, other factors regarding the memory that affect the learning process, such as adaptation rate, lower and upper bounds on the utility values are identified and tested using relatively short memory lengths as suggested before. Furthermore, two different heuristic selection strategies, based on the utility values, are assessed. Considering only the adaptation rates, the results support the previous

findings in Nareyek (2003). The combination of slow adaptation rates (additive/subtractive) seems to perform the best. The reinforcement learning – great deluge hyper-heuristic with the settings {lower bound=0, upper bound=40, heuristic selection strategy=*max*, positive adaptation rate =*additive*, negative adaptation rate=*subtractive*} improves the performance of the simple random – great deluge hyper-heuristic for solving the examination timetabling problem instances studied in this article.

The choice of reinforcement learning heuristic selection components affects the memory length which in turn also affects the intensification and diversification processes during the search. The set of low level heuristics contains neighbourhood operators that attempt to resolve conflicts due to a specific constraint type without considering whether the move will cause other constraint violations or not. The reinforcement heuristic selection chooses a constraint based heuristic as long as a given solution is improved in terms of overall quality. If the same heuristic starts generating worsening moves, the reinforcement learning heuristic selection method still supports the selection of the same heuristic until its utility value decreases to the same value as another one (or others). As soon as there is more than one low level heuristic with the same utility value, one of them is chosen randomly to give (another) chance to the other heuristic(s) for improving the candidate solution in hand. The intensification and diversification processes over the problem domain occur as a result of a dynamic interaction between heuristic selection, move acceptance and low level heuristics. The intensification process is activated whenever an improving move is accepted and continues as long as an improvement in the overall quality is achieved. Diversification arises in three ways which is mainly based on the great deluge move acceptance. Firstly, whenever there is a worsening move, the intensification phase ends. This worsening move might still be accepted by the great deluge allowing a jump to other promising regions of the search space. Secondly, a low level heuristic is included within the hyper-heuristic that perturbs a given candidate solution randomly using a small step size. If this heuristic is selected for invocation, it acts as a diversification mechanism. Finally, the random choice, in the case of equal utility values, provides an additional diversification mechanism. The reinforcement learning – great deluge hyper-heuristic with the given low level heuristics attempts to balance intensification and diversification automatically. It turns out to be successful in this attempt as this hyper-heuristic delivers a good performance over the Yeditepe University examination timetabling problem.

REFERENCES

Abdullah, S., Ahmadi, S., Burke, E. K., & Dror, M. (2007). Investigating Ahuja-Orlins large neighbourhood search for examination timetabling. *OR-Spektrum*, *29*(2), 351–372. doi:10.1007/s00291-006-0034-7

Alkan, A., & Özcan, E. (2003). Memetic algorithms for timetabling. In *Proceedings of 2003 IEEE Congress on Evolutionary Computation* (pp. 1796-1802).

Anagnostopoulos, A., Michel, L., Hentenryck, P. V., & Vergados, Y. (2006). A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, *9*, 177–193. doi:10.1007/s10951-006-7187-8

Ayob, M., & Kendall, G. (2003). A monte carlo hyper-Heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the International Conference on Intelligent Technologies (InTech'03)* (pp. 132-141).

Azimi, Z. N. (2005). Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, *163*(2), 705–733. doi:10.1016/j.amc.2003.10.061

Bai, R., Blazewicz, J., Burke, E., Kendall, G., & McCollum, B. (2007a). *A simulated annealing hyper-heuristic methodology for flexible decision support* (Computer Science Tech. Rep. No. NOTTCS-TR-2007-8). Nottingham, UK: University of Nottingham.

Bai, R., Burke, E. K., Kendall, G., & McCollum, B. (2007b). Memory length in hyper-heuristics: an empirical study. In *Proceedings of 2007 IEEE Symposium on Computational Intelligence in Scheduling (CISched2007)* (pp. 173-178).

Bai, R., & Kendall, G. (2003). An investigation of automated planograms using a simulated annealing based hyper-heuristics. In T. Ibaraki, K. Nonobe, & M. Yagiura (Ed.), *Meta-heuristics: Progress as Real Problem Solvers, selected papers from the 5th Metaheuristics International Conference (MIC'03)* (pp. 87-108). New York: Springer.

Bilgin, B., Özcan, E., & Korkmaz, E. E. (2007). An experimental study on hyper-heuristics and exam scheduling. In *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT'06)* (LNCS 3867, pp. 394-412).

Broder, S. (1964). Final examination scheduling. *Communications of the ACM*, *7*(8), 494–498. doi:10.1145/355586.364824

Burke, E., Bykov, Y., Newall, J. P., & Petrovic, S. (2003). A time-predefined approach to course timetabling. [YUJOR]. *Yugoslav Journal of Operations Research*, *13*(2), 139–151. doi:10.2298/YJOR0302139B

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Rong, Q. (2009b). *A survey of hyper-heuristics* (Computer Science Tech. Rep. No. NOTTCS-TR-SUB-0906241418-2747). Nottingham, UK: University of Nottingham.

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2009a). Exploring hyper-heuristic methodologies with genetic programming. In C. L. Mumford & L. C. Jain (Eds.), *Computational intelligence: Collaboration, fusion and emergence* (pp. 177-201). New York: Springer.

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2009c). *A classification of hyper-heuristic appraoches* (Computer Science Tech. Rep. No. NOTTCS-TR-SUB-0907061259-5808). Nottingham, UK: University of Nottingham.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenburg, S. (2003a). Hyper-heuristics: An emerging direction in modern search technology. In F. W. Glover & G. A. Kochenberger (Ed.), *Handbook of Metaheuristics* (Vol. 57, pp. 457-474). Dordrecht, The Netherlands: Kluwer International Publishing.

Burke, E., Kendall, G., & Soubeiga, E. (2003b). A tabu-search hyper-heuristic for timetabling and rostering. *Journal of Heuristics*, *9*, 451–470. doi:10.1023/B:HEUR.0000012446.94732.b6

Burke, E. K., Dror, M., Petrovic, S., & Qu, R. (2005). Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In B. L. Golden, S. Raghavan, & E. A. Wasil (Ed.), *The next wave in computing, optimization, and decision technologies: Proceedings of the 9th Informs Computing Society Conference* (pp. 79-91). Springer.

Burke, E. K., Elliman, D. G., Ford, P. H., & Weare, R. F. (1996a). Examination timetabling in British universities - a survey. In E. K. Burke & P. Ross (Eds.), *Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling,* Edinburgh (LNCS 1153, pp.76-92).

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, *176*(1), 177–192. doi:10.1016/j.ejor.2005.08.012

Burke, E. K., & Newall, J. P. (2004). Solving examination timetabling problems through adaption of heuristic orderings: Models and algorithms for planning and scheduling problems. *Annals of Operations Research*, *129*, 107–134. doi:10.1023/B:ANOR.0000030684.30824.08

Burke, E. K., Newall, J. P., & Weare, R. F. (1996b). A memetic algorithm for university exam timetabling. In E. K. Burke & P. Ross (Eds.), *Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling,* Edinburgh (LNCS 1153, pp. 241-250).

Burke, E. K., Petrovic, S., & Qu, R. (2006). Case based heuristic selection for timetabling problems. *Journal of Scheduling*, *9*, 115–132. doi:10.1007/s10951-006-6775-y

Caramia, M., & Dell'Olmo, P. (2007). Coupling stochastic and deterministic local search in examination timetabling. *Operations Research*, *55*(2). doi:10.1287/opre.1060.0354

Caramia, M., DellOlmo, P., & Italiano, G. F. (2001). New algorithms for examination timetabling. In S. Naher & D. Wagner (Eds.), *Algorithm Engineering 4th International Workshop (WAE'00)* (LNCS 1982, pp. 230-241).

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Operations Research*, *34*, 193–202. doi:10.1287/opre.34.2.193

Carter, M. W., & Laporte, G. (1996a). Recent developments in practical examination timetabling. In E. K. Burke & P. Ross (Eds.), *Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling,* Edinburgh (LNCS 1153, pp. 3-21).

Carter, M. W., Laporte, G., & Lee, S. (1996b). Examination timetabling: Algorithmic strategies and applications. *The Journal of the Operational Research Society*, *47*(3), 373–383.

Casey, S., & Thompson, J. (2003). GRASPing the examination scheduling problem. In E. K. Burke & P. De Causmaecker (Eds.), *Practice and theory of automated timetabling: Selected Papers from the 4th International Conference* (LNCS 2740, pp. 232-244).

Chakhlevitch, K., & Cowling, P. I. (2008). Hyperheuristics: Recent developments. In *Adaptive and Multilevel Metaheuristics* (pp. 3-29).

Chen, P.-C., Kendall, G., & Berghe, G. V. (2007). An ant based hyper-heuristic for the travelling tournament problem. In *Proceedings of IEEE Symposium of Computational Intelligence in Scheduling (CISched'07)* (pp. 19-26).

Cheong, C. Y., Tan, K. C., & Veeravalli, B. (2007). Solving the exam timetabling problem via a multi-objective evolutionary algorithm – a more general approach. In *Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched'07)* (pp. 165-172).

Cole, A. J. (1964). The preparation of examination timetables using a small-store computer. *The Computer Journal*, *7*, 117–121. doi:10.1093/comjnl/7.2.117

Corr, P. H., McCollum, B., McGreevy, M. A. J., & McMullan, P. (2006). A new neural network based construction heuristic for the examination timetabling problem. In T. P. Runarsson et al. (Eds.), *PPSN IX* (LNCS 4193, pp. 392-401).

Cowling, P., & Chakhlevitch, K. (2003). Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)* (pp. 1214-1221).

Cowling, P., Kendall, G., & Han, L. (2002). An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02)* (pp. 1185-1190).

Cowling, P., Kendall, G., & Soubeiga, E. (2001a). A hyperheuristic approach to scheduling a sales summit. In *Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling (PATAT'00)* (pp. 176-190). Springer-Verlag.

Cowling, P., Kendall, G., & Soubeiga, E. (2001b). A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of 4th Metaheuristics International Conference (MIC'01)* (pp. 127-131).

Cuesta-Canada, A., Garrido, L., & Terashima-Marin, H. (2005). Building hyper-heuristics through ant colony optimization for the 2d bin packing problem. In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'05)* (LNCS 3684, pp. 654-660).

Denzinger, J., Fuchs, M., & Fuchs, M. (1997). High performance ATP systems by combining several ai methods. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97)* (pp. 102-107).

Di Gaspero, L., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. In E. K. Burke & W. Erben (Eds.), *Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT'00)* (LNCS 2079, pp. 104-117).

Dowsland, K., & Thompson, J. (2005). Ant colony optimization for the examination scheduling problem. *The Journal of the Operational Research Society*, *56*(4), 426–438. doi:10.1057/palgrave.jors.2601830

Dowsland, K. A., Soubeiga, E., & Burke, E. (2007). A simulated annealing hyper-heuristic for determining shipper sizes. *European Journal of Operational Research*, *179*(3), 759–774. doi:10.1016/j.ejor.2005.03.058

Dueck, G. (1993). New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, *104*, 86–92. doi:10.1006/jcph.1993.1010

Erben, W. (2001). A grouping genetic algorithm for graph colouring and exam timetabling. In K. Burke & W. Erben (Eds.), *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT'00)* (LNCS 2079, pp. 132-156).

Ergul, A. (1996). GA-based examination scheduling experience at Middle East Technical University. In E. K. Burke & P. Ross (Eds.), *Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling,* Edinburgh (LNCS 1153, pp. 212-226).

Ersoy, E., Özcan, E., & Uyar, S. (2007). Memetic algorithms and hyperhill-climbers. In *Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA'07)* (pp. 156-166).

Even, S., Itai, A., & Shamir, A. (1976). On the complexity of timetable and multicommodity Flow problems. *SIAM Journal on Computing*, *5*(4), 691–703. doi:10.1137/0205048

Fisher, H., & Thompson, G. L. (1961). *Probabilistic learning combinations of local job-shop scheduling rules*. Paper presented at the *Factory Scheduling Conference,* Carnegie Institue of Technology.

Han, L., & Kendall, G. (2003). An investigation of a tabu assisted hyper-heuristic genetic algorithm. *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03), 3*, (pp. 2230-2237).

Kaelbling, L. P., Littman, M., & Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Keller, R. E., & Poli, R. (2007). Cost-benefit investigation of a genetic-programming hyper-heuristic. In *Proceedings of the 8th International Conference on Artificial Evolution (EA'07),* Tours, France (pp. 13-24).

Kendall, G., & Hussin, N. M. (2005). Tabu search hyper-heuristic approach to the examination timetabling problem at university of technology MARA. In E. K. Burke and M. Trick (Eds.), *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT'04)* (LNCS 3616, pp. 270-293).

Kendall, G., & Mohamad, M. (2004). Channel assignment in cellular communication using a great deluge hyper-heuristic. In *Proceedings of the 12th IEEE International Conference on Network (ICON'04)* (pp. 769-773).

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680. doi:10.1126/science.220.4598.671

Marin, H. T. (1998). *Combinations of GAs and CSP strategies for solving examination timetabling problems*. Unpublished PhD thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey.

Marín-Blázquez, J., & Schulenburg, S. (2005). A hyper-heuristic framework with XCS: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients. In T. Kovacs, X. Llorà, K. Takadama, P. Lanzi, W. Stolzmann, & S. Wilson (Eds.), *Proceedings of the 8th International Workshop on Learning Classifier Systems (IWLCS'05)* (LNCS 4399, pp. 193-218).

Merlot, L. T. G., Boland, N., Hughes, B. D., & Stuckey, P. J. (2002). A hybrid algorithm for the examination timetabling problem. In E. K. Burke & P. De Causmaecker (Eds.), *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT'02)* (LNCS 1153, pp. 207-231).

Nareyek, A. (2003). Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer decision-making* (pp. 523-544). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Ouelhadj, D., & Petrovic, S. (2008). A cooperative distributed hyper-heuristic framework for scheduling. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'08)* (pp. 2560-2565).

Özcan, E., Bilgin, B., & Korkmaz, E. E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, *12*, 3–23.

Özcan, E., Bykov, Y., Birben, M., & Burke, K. E. (2009). Examination timetabling using late acceptance hyper-heuristics. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC'09)* (pp. 997-1004).

Özcan, E., & Ersoy, E. (2005). Final exam scheduler – FES. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)* (Vol. 2, pp. 1356-1363).

Paquete, L. F., & Fonseca, C. M. (2001). A study of examination timetabling with multiobjective evolutionary algorithms. In *Proceedings of the 4th Metaheuristics International Conference (MIC'01)* (pp. 149-154).

Petrovic, S., Patel, V., & Yang, Y. (2005). Examination timetabling with fuzzy constraints. In E. K. Burke & M. Trick (Eds.), *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT'05)* (LNCS 3616, pp. 313-333), Springer.

Petrovic, S., Yang, Y., & Dror, M. (2007). Case-based selection of initialisation heuristics for metaheuristic examination timetabling. *Expert Systems with Applications: An International Journal*, *33*(3), 772–785. doi:10.1016/j.eswa.2006.06.017

Pillay, N., & Banzhaf, W. (2008). A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. *European Journal of Operational Research*. doi:. doi:10.1016/j.ejor.2008.07.023

Qu, R., Burke, E. K., & McCollum, B. (2008). Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, *198*(2), 392–404. doi:10.1016/j.ejor.2008.10.001

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*(1), 55–89. doi:10.1007/s10951-008-0077-5

Rattadilok, P., Gaw, A., & Kwan, R. (2005). Distributed choice function hyper-heuristics for timetabling and scheduling. In *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT'2004)* (pp. 51-67).

Ross, P. (2005). Hyper-heuristics. In: E. K. Burke & G. Kendall (Eds.), *Search methodologies: Introductory tutorials in optimization and decision support techniques* (Ch. 17, pp. 529-556). New York: Springer.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Terashima-Marin, H. T., Moran-Saavedra, A., & Ross, P. (2005). Forming hyper-heuristics with GAs when solving 2D-regular cutting stock problems. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation* (Vol. 2, pp. 1104-1110).

Terashima-Marin, H. T., Ross, P., & Valenzuela-Rendon, M. (1999). Evolution of constraint satisfaction strategies in examination timetabling. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99)* (pp. 635-642).

Tereshima-Marin, H. T., Zarate, C. J. F., Ross, P., & Valenzuela-Rendon, M. (2007). Comparing two models to generate hyper-heuristics for the 2d-regular bin-packing problem. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)* (pp. 2182-2189).

Thompson, J. M., & Dowsland, K. A. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, *25*, 637–648. doi:10.1016/S0305-0548(97)00101-9

Vazquez-Rodriguez, J. A., Petrovic, S., & Salhi, A. (2007). A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup times and uniform machines. In P. Baptiste, G. Kendall, A. Munier-Kordon, & F. Sourd (Eds.), *Proceedings of the 3rd Multi-disciplinary International Scheduling Conference: Theory and Applications (MISTA'07),* Paris (pp. 506-513).

Wong, T., Cote, P., & Gely, P. (2002). Final exam timetabling: a practical approach. In . *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering*, *2*, 726–731.

Ender Ozcan is a science and innovation lecturer with the Automated Scheduling, Optimisation and Planning (ASAP)Research Group in the School of Computer Science at the University of Nottingham, UK. He received his PhD from the Department of Computer and Information Science at Syracuse University, NY, USA in 1998. He worked as a lecturer in the Department of Computer Engineering at Yeditepe University, Istanbul, Turkey from 1998-2007. He established and led the ARTIficial Intelligence research group from 2002 and awarded two research grants from TUBITAK. He served as the deputy head of the Department from 2004-2007. Dr. Ozcan joined the ASAP group as a senior research fellow in 2008. He has been serving as an executive committee member for the LANCS initiative, which is one of the largest Science and Innovation Rewards given by EPSRC (Engineering and Physical Sciences Research Council, UK). His research interests and

activities lie at the interface of computer science and operational research. He has been leading studies in the field of metaheuristics focusing on evolutionary algorithms (memetic algorithms, PSO), hyper-heuristics, and their applications to the real-world and theoretical problems. Dr. Ozcan has published over 55 refereed papers. He has been a member of the program committees in major international conferences and refereeing for reputable journals. He has co-organised five workshops on hyper-heuristics and metaheuristics and he is the guest co-editor of the forthcoming first special issue on hyper-heuristics (Journal of Heuristics, 2009).

Mustafa Misir received his BSc and MSc degrees in computer engineering from Yeditepe University in 2007 and 2008, respectively. He was awarded as a student assistant during his undergraduate study. He worked as a teaching and research assistant during his graduate study in the Artificial Intelligence research group at Yeditepe University. Currently, he is studying towards PhD in the Department of Computer Science (Informatics) at Katholieke Universiteit Leuven as a research fellow. He is also working as a researcher in the CODeS Research Group, Katholieke Universiteit Leuven, Campus Kortrijk and IT Research Group, KaHo Sint-Lieven. His research interests include combinatorial optimization, hyper-heuristics, meta-heuristics and reinforcement learning.

Gabriela Ochoa is a senior research fellow with the Automated Scheduling, Optimisation and Planning (ASAP) Research Group in the School of Computer Science at the University of Nottingham, since October 2006, where she coordinates a project on 'Automated Heuristic Design'. She received her PhD in computer science and artificial intelligence from the University of Sussex, UK, in 2001. Gabriela Ochoa has been involved with inter-disciplinary research, and foundations and applications of evolutionary algorithms since the mid 90s, and more recently with meta-heuristics and hyper-heuristics. She has published over 30 refereed research articles, serves on the program committees of major conferences in evolutionary computation and meta-heuristics, and has refereed for reputable journals in these fields. Ochoa has recently dictated a tutorial on hyper-heuristics, and proposed and co-organised two workshops on hyper-heuristic methodologies held as part of reputable evolutionary computation conferences. She is the guest co-editor of the first special issue on hyper-heuristics (Journal of Heuristics, 2009).

Edmund K. Burke is dean of the Faculty of Science at the University of Nottingham and he leads the Automated Scheduling, Optimisation and Planning (ASAP) Research Group in the School of Computer Science. He is a member of the EPSRC Strategic Advisory Team for Mathematics. He is a fellow of the Operational Research Society and the British Computer Society and he is a member of the UK Computing Research Committee (UKCRC). Prof. Burke is editor-in-chief of the Journal of Scheduling*, area editor (for combinatorial optimisation) of the* Journal of Heuristics*, associate editor of the* INFORMS Journal on Computing*, associate editor of the* IEEE Transactions on Evolutionary Computation *and a member of the editorial board of* Memetic Computing. *He is also the research director of EventMAP Ltd. and a director of Aptia Solutions Ltd, both of which are spin out companies from the ASAP group. Prof. Burke has played a leading role in the organisation of several major international conferences in his research field in the last few years. He has edited/authored 14 books and has published over 180 refereed papers. He has been awarded 47 externally funded grants worth over £11M from a variety of sources including EPSRC, ESRC, BBSRC, EU, Research Council of Norway, East Midlands Development Agency, HEFCE, Teaching Company Directorate, Joint Information Systems Committee of the HEFCs and commercial organisations. This funding portfolio includes being the principal investigator on a recently awarded EPSRC Science and Innovation award of £2M, an EPSRC grant of £2.6M to investigate the automation of the heuristic design process and an EPSRC platform grant worth £423K.*