Ann Oper Res
DOI 10.1007/s10479-010-0782-2

Monte Carlo hyper-heuristics for examination timetabling

Edmund K. Burke · Graham Kendall · Mustafa Mısır · Ender Özcan

Abstract Automating the neighbourhood selection process in an iterative approach that uses multiple heuristics is not a trivial task. Hyper-heuristics are search methodologies that not only aim to provide a general framework for solving problem instances at different difficulty levels in a given domain, but a key goal is also to extend the level of generality so that different problems from different domains can also be solved. Indeed, a major challenge is to explore how the heuristic design process might be automated. Almost all existing iterative selection hyper-heuristics performing single point search contain two successive stages; heuristic selection and move acceptance. Different operators can be used in either of the stages. Recent studies explore ways of introducing learning mechanisms into the search process for improving the performance of hyper-heuristics. In this study, a broad empirical analysis is performed comparing Monte Carlo based hyper-heuristics for solving capacitated examination timetabling problems. One of these hyper-heuristics is an approach that overlaps two stages and presents them in a single algorithmic body. A learning heuristic selection method (L) operates in harmony with a simulated annealing move acceptance method using reheating (SA) based on some shared variables. Yet, the heuristic selection and move acceptance methods can be separated as the proposed approach respects the common selection hyper-heuristic framework. The experimental results show that simulated annealing with reheating as a hyper-heuristic move acceptance method has significant potential. On the other hand, the learning hyper-heuristic using simulated annealing with reheating move

E.K. Burke · G. Kendall · E. Özcan (✉)
Automated Scheduling, Optimisation and Planning Research Group, School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
e-mail: exo@cs.nott.ac.uk

E.K. Burke
e-mail: ekb@cs.nott.ac.uk

G. Kendall
e-mail: gxk@cs.nott.ac.uk

M. Mısır
Department of Computer Engineering, Yeditepe University, Inonu Mahallesi, Kayisdagi Caddesi, Kadikoy, Istanbul 34755, Turkey
e-mail: mmisir@cse.yeditepe.edu.tr

Published online: 05 September 2010

acceptance (L–SA) performs poorly due to certain weaknesses, such as the choice of rewarding mechanism and the evaluation of utility values for heuristic selection as compared to some other hyper-heuristics in examination timetabling. Trials with other heuristic selection methods confirm that the best alternative for the simulated annealing with reheating move acceptance for examination timetabling is a previously proposed strategy known as the choice function.

Keywords Hyper-heuristics · Simulated annealing · Meta-heuristics · Examination timetabling · Reinforcement learning

1 Introduction

One definition of a hyper-heuristic is a (meta-)heuristic that carries out a search over the heuristic space formed by a set of low level heuristics (Burke et al. 2003). Low level heuristics can be either selected or generated during the search process. For example, Burke et al. (2006, 2007a) uses a Genetic Programming meta-heuristic as a hyper-heuristic to search for the best heuristic for solving a given online bin packing problem. During the evolutionary search process, a heuristic is constructed using indivisible program components via genetic operators. On the other hand, selection hyper-heuristics which manage a set of perturbative low level heuristics, utilising a single configuration during the search, are usually iterative methods (Bilgin et al. 2007; Özcan et al. 2006, 2008).

A selection hyper-heuristic framework is provided in Fig. 1. At each iteration, the most suitable heuristic (or a subset) is chosen using a heuristic selection method and a new state is generated after the application of the selected heuristic(s). This move is either accepted or rejected based on an *acceptance method*. The process continues until a termination criterion is met. A selection hyper-heuristic will be denoted as *heuristic selection method–move acceptance method* from this point onward. At the high level, a hyper-heuristic interacts with the problem domain via low level heuristics and gathers problem independent information such as the number of heuristics, the quality change in a candidate solution after applying a selected heuristic, or the success of a heuristic. Two major components are identified in existing selection hyper-heuristics: heuristic selection and move acceptance. Cowling et al. (2001a) used this framework and compared the performance of a number of *simple* hyper-heuristics embedding different heuristic selection components over a scheduling problem: *Simple Random*, *Random Permutation*, *Random Descent*, *Random Permutation Descent*, *Greedy* and *Choice Function*. The authors employed only two simple acceptance methods in their study: (i) all moves are accepted (AM), and (ii) only improving moves are accepted (OI). According to the experimental results, the Choice Function–All Moves Accepted hyper-heuristic showed potential.

Simulated annealing, proposed by Kirkpatrick et al. (1983) and Cerny (1985) independently, is a very well-known meta-heuristic within the operational research and artificial intelligence communities. A comparative study by Bilgin et al. (2007) shows that combining a different heuristic selection method with a different acceptance method might yield improved performance. Furthermore, empirical results indicate that Monte Carlo based hyper-heuristics, specifically the ones utilising simulated annealing deliver superior performances. Similarly, Bai and Kendall (2005), Dowsland et al. (2007) and Bai et al. (2007) report that variants of simulated annealing have great potential as a move acceptance component in hyper-heuristics. Hence, this study investigates different selection hyper-heuristics that utilise stochastic Monte Carlo based move acceptance methods for solving examination timetabling problems.

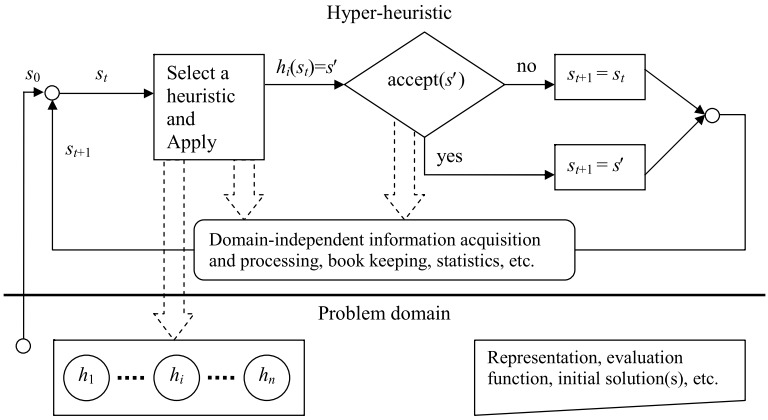

Fig. 1 A hyper-heuristic framework

Section 2 summarises previous studies on examination timetabling and describes the problem formulation used in this study. Section 3 presents the details of selection hyper-heuristics for examination timetabling used during the experiments. The computational results are discussed in Sect. 4. Finally, conclusions are provided in Sect. 5.

2 The examination timetabling problem

Most real-world examination timetabling problems can be represented as a constraint optimisation problem using a 3-tuple $\langle E, I, K \rangle$, where E represents events (variables), namely examinations to be scheduled in this work, I contains the domain for each event and K is the constraints set. A domain can be a product of sets, each representing a different resource to be allocated for a given event. In this study, only time is considered as a resource.

$$E = \{e_1, e_2, \ldots, e_N\} \tag{1}$$

$$K = \{k_1, k_2, \ldots, k_M\} \tag{2}$$

Let $S = \{s_1, s_2, \ldots, s_L\}$ denote possible ordered list of start times for an examination. The examination timetabling problem can be described as a search for the best assignment $\forall x, \exists y$ $(e_x = s_y)$, where $e_x \in E$ and $s_y \in S$, such that, given constraints are satisfied. The assignment implies that the examination e_x starts at s_y.

Two different types of constraints are identified: *hard* and *soft* constraints. Hard constraints must be satisfied, while soft constraints should be respected as much as possible. The size of the search space for a timetabling problem increases exponentially as the number of items to be scheduled increases and they are known to be NP-complete constraint optimisation problems (Even et al. 1976). Hence, an optimal solution might not be obtained by a traditional approach. Many researchers have been applying many different non-traditional methodologies to solve many different types of timetabling problems.

2.1 An overview of approaches to examination timetabling

The initial studies on computer based strategies for examination timetabling date back to the 1960s (Cole 1964; Broder 1964). Research interest in examination timetabling has been

progressively increasing since then. Carter et al. (1996) applied graph colouring heuristics to construct examination timetables based on the fact that a basic version of a timetabling problem can be reduced to a graph colouring problem (Leighton 1979). Moreover, Carter provided a set of benchmark problem instances widely used by the examination timetabling community, referred to as the *Toronto benchmarks*. In addition to heuristics (e.g., Marin 1998; Burke and Newall 2004), a variety of approaches are used for addressing a variety of examination timetabling problems.

Evolutionary algorithms (Goldberg 1989; Ong et al. 2006; Krasnogor and Gustafson 2004) are the most frequently used approaches for examination timetabling. Burke et al. (1996a) applied memetic algorithms for solving a subset of Toronto benchmarks and Nottingham University problems. Ergul (1996) implemented a steady state genetic algorithm for solving the examination timetabling problem at the Middle East Technical University, Ankara. Paquete and Fonseca (2001) designed a multi-objective evolutionary algorithm, where each objective aims to satisfy a different type of constraint. Wong et al. (2002) used a memetic algorithm with a non-elitist replacement strategy to solve an examination timetabling problem at École de Technologie Supérieure. Burke and Newall (1999) combined a problem decomposition strategy with a memetic algorithm for incrementally solving large examination timetabling problems. This strategy was modified later by Özcan and Alkan (2007). Özcan and Ersoy (2005) proposed a framework for designing violation directed adaptive operators, inspired from the previous studies in Ross et al. (1994), Corne et al. (1994) and Alkan and Özcan (2003). Ersoy et al. (2007) investigated a set of hyper-heuristics for selecting a hill climber during the evolutionary process for solving examination timetabling problems.

In addition to the hyper-heuristics based on constructive low level heuristics (Kendall and Hussin 2005; Burke et al. 2007b), tabu search (Gaspero and Schaerf 2001), very large neighbourhood search (Abdullah et al. 2007), simulated annealing (Merlot et al. 2003), multi-stage approaches utilising case based reasoning (Petrovic et al. 2007), an iterative greedy algorithm (Caramia et al. 2001, 2008) and great deluge (Müller 2009), fuzzy reasoning (Petrovic and Patel 2005; Asmuni et al. 2005), neural network (Corr et al. 2006) and ant colony optimisation (Dowsland and Thompson 2005; Eley 2006) based approaches and hybrid methods (Azimi 2005; Gogos et al. 2010) are some of the other techniques used to solve different types of examination timetabling problems.

It is always interesting to know the state of the art approach for examination timetabling. Unfortunately, there are many variants which make it difficult to compare different methodologies. Schaerf (2006) emphasise the importance of comparability and reproducibility of the results in the research community. Competitions, such as ITC2007 (McCollum et al. 2010), take an active role in setting the state of the art for different problems including examination timetabling. Approaches compete in a fair environment over a set problems. It is vital for researchers to generate robust and flexible approaches that will be able to solve unseen examination timetabling problem instances for a given institution. On the other hand, there are many other issues that have to be dealt with considering the timetabling users. For example, an easy to use graphical interface is as important as the embedded state of the art approach for solving the problem itself from the user point of view. McCollum (2006) discusses real world issues in examination and course timetabling observing that there is a gap between research and practice from a commercial point of view. More on examination timetabling can be found in the survey provided by Qu et al. (2009) which updates Carter and Laporte (1996) and Carter (1986). The wider timetabling literature is discussed in more detail in Burke et al. (1996b), Schaerf (1999) and Burke and Petrovic (2002).

2.2 Problem formulation

Investigation of selection hyper-heuristics for examination timetabling is still an open research area. In this study, a set of stochastic Monte Carlo based hyper-heuristics is compared for solving a real-world problem as described in Bilgin et al. (2007). An examination starts and ends within a given time slot of three hours and has to be assigned to a single time slot.

$$X_{ij} = 1 \quad \text{if } e_i \text{ starts at } s_j,\ 0 \text{ otherwise} \tag{3}$$

$$\forall i, e_i \in E. \sum_{\forall j, s_j \in S} X_{ij} = 1 \tag{4}$$

This problem requires that the following constraints be satisfied:

– Examination conflict (k_1): A student cannot sit for more than one examination at any given time.

$$r_{ti} : 1 \text{ if the } t\text{th student takes the examination } e_i \in E,\ 0 \text{ otherwise}$$

$$\forall t. \forall j, s_j \in S. \sum_{\forall i, e_i \in E} X_{ij} r_{ti} \leq 1 \tag{5}$$

– Seating restriction (k_2): The total number of students seated for all examinations at a given time slot cannot exceed the pre-determined capacity (C).

$$b_i = \sum_{\forall t} r_{ti} \tag{6}$$

$$\forall j, s_j \in S. \sum_{\forall i, e_i \in E} X_{ij} b_i < C \tag{7}$$

Successive examination restriction (k_3): It is also *strongly* preferable that there is a single time slot between two successive examinations of a student in the same day. Since the size of the gap is not important, disallowing successive examination assignments for a student in the same day resolves this preference.

$$o_{uv} : 1 \text{ if } s_u, s_v \in S \text{ are in the same day, 0 otherwise}$$

$$\forall t. \sum_{\substack{\forall i, e_i \in E \\ \forall j, e_j \in E \\ i \neq j}} \sum_{\forall u \neq L, s_u \in S} r_{ti} r_{tj} X_{iu} X_{j(u+1)} o_{u(u+1)} = 0 \tag{8}$$

The evaluation function measures the quality of a given solution T based on the weighted sum (w_i) of the number of these three types of constraint violations ($viol(k_i, T)$) given a set of students and the examinations that they have registered. If all the constraints are respected then the evaluation function returns -1.

$$evaluate(T) = \frac{-1}{\sum_{\forall i, k_i \in K} w_i viol(k_i, T) + 1} \tag{9}$$

Fig. 2 Pseudocode of Monte Carlo based hyper-heuristic

```
MONTE_CARLO_HYPER_HEURISTIC()
Input: initial candidate solution s0 with a quality of f0
Output: final candidate solution; sfinal, ffinal
1.  Initialisation: i=0, H, P, etc.
2.  while ( i < maxIterations ) {
3.      // heuristic selection
4.      snew = selectApplyHeuristic(P, H, si);
5.      update0(P);
6.      fnew = evaluate(snew);
7.      update1(P);
8.      // move acceptance
9.      Δf = fnew − fi;
10.     if (Δf≤0) then
11.         si+1 = snew; fi+1 = fnew; update2(P);
12.     else
13.         if ( (Δf<0) and PD(P)<rand(0,1) ) then
14.             si+1 = snew; fi+1 = fnew; update3(P);
15.     i++;
16.     update4(P);
17. }
18. return sfinal, ffinal
```

3 Selection hyper-heuristics for examination timetabling

3.1 A Monte Carlo based hyper-heuristic framework

Figure 2 shows the Monte Carlo hyper-heuristic framework used in this study. The heuristic selection method decides which low level perturbative heuristic (h_{ID}) to apply to the candidate solution at hand. If the solution improves or its quality stays the same ($\Delta f \leq 0$, assuming a minimisation problem), then it is accepted. If the solution worsens, it can still be accepted with a probability based on a function, PD. In addition, the hyper-heuristic maintains problem independent information. In Fig. 2, line 1, H is the pool of low level heuristics. P represents the data structure(s) holding all relevant problem domain independent information that might be required by a Monte Carlo hyper-heuristic, such as statistics, or a utility value for each low level heuristic. An update function (labelled from update0-4) either does nothing or keeps track of some relevant information during the search process depending on the components of a Monte Carlo hyper-heuristic. An update function is assumed to have access to the local variables, such as i, s_i, f_{new} or Δf, since they might be used as a parameter within a Monte Carlo hyper-heuristic component at a given iteration. For example, PD might require Δf.

3.2 Heuristic selection methods

Simple Random (SR), Greedy (GR), Choice Function (CF) and a learning scheme (L) are utilised as heuristic selection methods. Simple Random selects a heuristic randomly with equal probability. The Greedy method tests all heuristics using the same candidate solution. It keeps the new solution which has the best quality and feeds it into the move acceptance. The choice function maintains a record of the performance of each heuristic. Three different criteria are maintained: i) the individual performance of the heuristic (10), ii) how well

the heuristic has performed with other heuristics (11) and iii) the elapsed time since the heuristic has been called (12). The heuristic having the best score is selected at each step and applied to the candidate solution. If h_{ID} is the selected low level heuristic, then the scores of all heuristics are updated using (13). Cowling et al. (2001a, 2002) tested these heuristic selection methods against AM and OI move acceptance methods. The CF parameters α, β ($\in (0, 1]$) and δ are adjusted automatically as described in Cowling et al. (2001b). They emphasise the importance of the latest performance.

$I_n(y)$ and $T_n(y)$ ($I_n(x, y)$ and $T_n(x, y)$) denote the change in the evaluation function and the amount of time taken, respectively, when the nth last time the heuristic y was selected and employed (immediately after the heuristic x).

$$\forall i,\ g_1(h_i) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \tag{10}$$

$$\forall i,\ g_2(h_{\mathrm{ID}}, h_i) = \sum_n \beta^{n-1} \frac{I_n(h_j, h_{\mathrm{ID}})}{T_n(h_j, h_{\mathrm{ID}})} \tag{11}$$

$$\forall i,\ g_3(h_i) = elapsedTime(h_i) \tag{12}$$

$$\forall i,\ score(h_i) = \alpha g_1(h_i) + \beta g_2(h_{\mathrm{ID}}, h_i) + \delta g_3(h_i) \tag{13}$$

3.3 Move acceptance methods

Ayob and Kendall (2003) report a fast move acceptance method which is similar to simulated annealing (Kirkpatrick et al. 1983; Cerny 1985). The proposed hyper-heuristic fits into the framework presented in Fig. 2. It is referred to as EMCQ and the approach uses (14) as its PD function. The only update required is incrementing a counter whenever a worsening move occurs and resetting it whenever there is an improvement in the solution quality. The authors tested only the Simple Random heuristic selection method in their study.

$$e^{-\frac{\Delta f m}{Q}}, \tag{14}$$

Q in (14) is a counter for successive worsening moves.

m is the unit time in minutes that measures the duration of the heuristic execution.

Considering that different machines have different properties, EMCQ cannot be thought of as a general strategy. Obviously, more instructions will be executed in a unit time on a fast machine as compared to a slower machine. In this study, m counts the number of successive B steps, e.g., m is incremented at every 200 iterations. This type of strategy generates a more general acceptance method applicable in different environments.

Özcan et al. (2006) compared the performances of many heuristic selection and move acceptance combinations in hyper-heuristics. The results show that a standard simulated annealing move acceptance performs the best, especially combined with the Choice Function approach. Simulated annealing uses a linear cooling schedule as shown in (15). This method will be denoted as MC, respecting their notation.

$$e^{-\frac{\Delta f}{\Delta F(1-\frac{i}{T})}}, \tag{15}$$

ΔF in (15) is an expected range for the maximum fitness change and $T = maxIterations$. Since the number of violation would be 0 for the best case, ΔF is computed by evaluating the first configuration.

Bai and Kendall (2005) showed that a more elaborate simulated annealing hyper-heuristic based on Metropolis criterion is also promising. Bai et al. (2007) extended this previous study and presented a new hyper-heuristic scheme that embeds a variant of the reinforcement learning mechanism (L) into the heuristic selection process. The move acceptance method employs a more sophisticated scheme using annealing and reheating phases (SA). SA is a modified version of MC with reheating. The proposed hyper-heuristic is denoted as L–SA respecting their notation. During the annealing phase, the following formula is used as the PD function (in Fig. 4, line 13):

$$e^{-\frac{\Delta f}{t}} \tag{16}$$

The temperature (t) is reduced based on the nonlinear function provided by Lundy and Mees (1986):

$$t = \frac{t}{1+\beta t}, \tag{17}$$

$$\beta = \frac{(t_0 - t_{final})i_{temp}}{maxIterations \cdot t_0 \cdot t_{final}}, \tag{18}$$

i_{temp} is the number of iterations at a temperature

During the reheating phase, $t = \frac{t}{1-\beta t}$ is used to increase the temperature up to the temperature when the last improvement was observed. Then the system goes into the annealing phase again. In their study, Bai et al. (2007) present a learning heuristic selection method and SA as a united framework. In this study, heuristic selection and the move acceptance components of a hyper-heuristic are separated in order to evaluate the performance of SA with reheating as suggested using different heuristic selection methods. The learning mechanism for heuristic selection, denoted as L, assigns a weight to each heuristic and updates them periodically. The percentage of accepted calls and the calls generating new solutions made to a heuristic are used as the weight of a given heuristic during the annealing and reheating phases, respectively. These weights are then used to select a heuristic based on a random choice strategy. The L–SA hyper-heuristic is tested over nurse rostering, course timetabling and bin packing problems using nine, three and five low level heuristics. The results demonstrate the success of L–SA. More on the algorithmic details and parametric choices can be found in Bai et al. (2007).

In this study, three Monte Carlo based move acceptance methods described above are utilised: standard simulated annealing (MC), simulated annealing with reheating (SA) and exponential Monte Carlo (EMCQ). EMCQ differs from simulated annealing, since it does not require any cooling schedule.

3.4 Representation and low level heuristics

A candidate solution is implemented as an ordered array of integer values representing a time-slot assigned for each exam. This direct encoding is illustrated in Fig. 3. Four heuristics are implemented to be used with the hyper-heuristics for solving an examination timetabling problem. Figure 4 shows the pseudo-code of these heuristics. Three of them perform a search over constraint based neighbourhoods based on a *tournament* strategy employed while selecting an examination for rescheduling and assigning a new period. The number of items (e.g. examinations) that are allowed to enter into a tournament is referred to as *tour-size*. In our implementation, none of the selected items are allowed to be the same item. Hence,

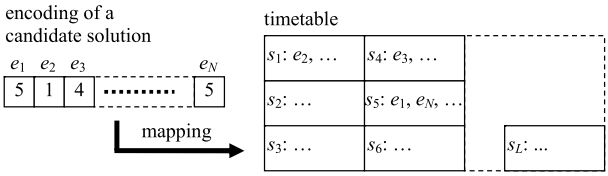

Fig. 3 Encoding of a candidate solution and its mapping (decoding)

– *RESCHEDULE_EXAM_CONFLICT – RESCHEDULE_ SUCCESSIVE_EXAM*

1. Choose *tour-size*1 number of examinations for tournament, randomly.
2. Determine the *winner* examination for rescheduling that generates maximum number of conflicts for the given constraint. In case of equality, employ random selection.
3. Choose *tour-size*2 number of time slots for tournament, randomly.
4. Assign the selected examination to the *winner* time slot that has the least number of targeted constraint type violations. In case of equality, employ random selection.

– *RESCHEDULE_SEATING*

1. Choose *tour-size*3 number of time slots for tournament, randomly. This phase guarantees that at least one of the selected time slots contains at least one examination assignment.
2. Determine the one (winner) with the maximum capacity conflict.
3. Choose *tour-size*4 number of examinations for tournament, randomly that are assigned to the selected time slot.
4. Determine the *winner* examination that has more attendants for rescheduling. In case of equality, employ random selection.
5. Choose *tour-size*5 number of time slots for tournament, randomly.
6. Assign the selected examination to the *winner* time slot that has the least seating capacity violation. In case of equality, employ random selection.

– *RANDOM_RESCHEDULING*

Make a pass over the examinations one by one and randomly reschedule an examination with a probability of $1/number\_of\_exams$

Fig. 4 Pseudocode of low level heuristics

given I items, the probability of the wth item to be selected for tournament is $w/(I-w+1)$, where $1 \leq w \leq tour\text{-}size$. A single item wins the tournament based on a predetermined criterion. It is allowed to select a number of items for tournament less than *tour-size*, since it might not be possible to find unique *tour-size* number of items based on the criterion at a given time. The last heuristic is a random perturbation similar to the *mutation* operator in Genetic Algorithms (Goldberg 1989).

A constraint based heuristic considers only a specific constraint type and ignores the rest during the rescheduling of a selected exam. Only the violations caused by a targeted constraint type are attempted to be reduced. Improvement after such a process does not imply that the overall quality of a candidate solution will improve as well. On the contrary, a new candidate solution with a worse quality value might arise. This is the reason why the move acceptance method becomes important within the framework of selection hyper-heuristics based on perturbative low level heuristics.

Table 1 Characteristics of the modified Toronto benchmark dataset used during the experiments

Instance	Examinations	Students	Enrollment	Conflict density	Days	Capacity
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

4 Computational experiments

4.1 Experimental setup

Carter et al. (1996) provided the most commonly used data set, referred to as the Toronto benchmark set by the examination timetabling community. There are 13 real world problems in the data set. 10 of them are obtained from the universities around the world and the rest from Canadian high schools. We use the standard notation introduced by Qu et al. (2009). Although there are similarities with the previous studies, the examination timetabling problem described in this work is unique. Initially, it is presented by Özcan and Ersoy (2005) as a real world problem that is dealt with at Yeditepe University, Istanbul. The Toronto benchmark data is extended with new properties, accordingly and used in Bilgin et al. (2007). This problem is a capacitated variant of examination timetabling. There is a maximum *capacity* of seating available during exams at each time slot. The timetable size is fixed with three examination slots per day for a given number of days. The characteristics of each problem instance based on the formulation in Sect. 2.2 are summarised in Table 1.

Pentium IV, 3 GHz Linux machines with 2 Gb memory are used during the experiments. Fifty runs are performed with each hyper-heuristic for a given problem instance. Fifty initial configurations are generated randomly for each problem instance and the same set of fifty initial configurations are used by each hyper-heuristic for each problem instance during the runs. Bai et al. (2007) argue that the L–SA hyper-heuristic performs better as the number of iterations is increased. Hence, all the approaches are tested using two termination criteria using two values for the maximum number of iterations is exceeded; 10^6 and 10^7. Only one evaluation is performed at each iteration.

As a performance criterion, ranking based on the best result from the runs is used. Ranks are in the range [1..*number-of-hyper-heuristics*] from best to worst and ties are considered. As another performance measure, *%-improvement* is used as in (19).

$$\%\text{-}improvement(x, y) = \frac{Q_x - Q_y}{Q_y}, \tag{19}$$

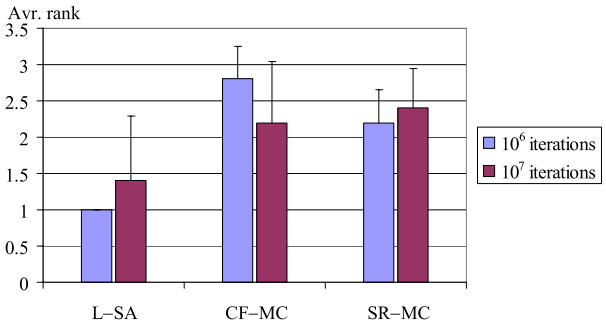

Fig. 5 Average ranks with standard deviations of SR–MC, CF–MC and L–SA hyper-heuristics for solving hecs92 I, ear83 I, tre92, lse91 and car91 I

where Q_x denotes the quality of the best result obtained in fifty runs using the hyper-heuristic x. Hence, this value shows how much the approach x improves over the best result produced by y. Similarly, *average %-improvement* computes the percentage improvement introduced by an approach over another one using the average quality of solutions over fifty runs for a given problem instance. The *cumulative distribution function* as described in (20) is obtained based on the experiments.

$$F(x) = P(X \leq x) = \sum_{x_i \leq x} p(x_i) \tag{20}$$

where $i \in [1..50]$, X is a discrete random variable, x_i is the quality of the best solution obtained in the ith run, and $p(x_i)$ is the probability to attain x_i. In the following sections, our computational results are discussed.

4.2 Comparison of L–SA, SR–MC and CF–MC

The reinforcement learning–simulated annealing with reheating hyper-heuristic (L–SA) has not been tested on the examination timetabling problems before (Bai et al. 2007). On the other hand, Bilgin et al. (2007) show that the Choice Function (CF) heuristic selection combined with standard simulated annealing (denoted as MC) performs the best for examination timetabling. This hyper-heuristic is one of the best known approaches reported so far for solving the examination timetabling problems in Table 1. A set of preliminary experiments are performed to compare L–SA to SR–MC and CF–MC over a subset of benchmark problems, namely; hecs92 I, ear83 I, tre92, lse91 and car91 I. In the previous study, hyper-heuristics are given 600 seconds to execute. In this study, the termination criterion is the number of evaluations. Hence, the experiments using MC hyper-heuristics are repeated. Considering the best in fifty runs, the performances of the hyper-heuristics are ranked from 1 to 3 for each problem instance. Figure 5 illustrates the average rank of each hyper-heuristic for a given maximum number of iterations.

L–SA is superior to the other hyper-heuristics based on the standard simulated annealing in finding the best. When the best performance of SR–MC is compared to CF–MC, it is observed that SR–MC is slightly better than CF–MC for the maximum number of iterations of 10^6. Choice Function beats the simple random heuristic selection when the maximum number of iterations is increased to 10^7. It seems that given enough time, the learning mechanism within the Choice Function becomes more effective in generating high quality solutions. L–SA performs consistently the best for each problem instance when the number of steps is increased.

Table 2 %–improvement generated by L–SA over CF–EMCQ and SR–EMCQ for each problem instance as the maximum number of iterations change from 10^6 to 10^7. Bold entries compare and mark the best hyper-heuristic from CF–EMCQ and SR–EMCQ for a given problem instance and termination criterion

	L–SA %–improvement			
	maxIteration $= 10^6$		maxIteration $= 10^7$	
problem	CF–EMCQ	SR–EMCQ	CF–EMCQ	SR–EMCQ
car91 I	96.83	**96.80**	98.55	**86.06**
car92 I	88.64	**88.48**	**90.58**	90.70
ear83 I	**73.60**	74.71	**78.68**	79.92
hecs92 I	**83.84**	84.44	84.50	**84.45**
kfu93	**95.25**	95.36	**95.24**	95.29
lse91	**79.85**	80.30	**83.20**	84.04
pur93 I	**81.36**	81.49	**86.11**	86.65
rye92	**93.43**	93.46	**93.77**	94.15
sta83 I	**52.09**	54.21	**50.62**	52.19
tre92	92.27	**91.99**	92.74	**92.35**
uta92 I	**92.09**	92.44	**93.61**	94.08
ute92	**56.38**	57.86	58.44	**58.34**
yor83 I	**73.75**	75.25	**76.46**	77.29

The average performance comparison of these hyper-heuristics yields the same results and reveals the success of L–SA. CF–MC delivers a similar average performance to SR–MC. The Choice Function is slightly better than Simple Random when combined with MC for maxIteration $= 10^7$, and vice versa for maxIteration $= 10^6$. Excluding ear83 I, L–SA generates an average %-improvement of at least 32% over CF–MC for each problem instance regardless of the termination criterion, hence L–SA and CF–MC are kept for further experiments.

4.3 Performance of the EMCQ based hyper-heuristics

Ayob and Kendall (2003) concludes that the Simple Random–EMCQ hyper-heuristic is the best choice for solving component placement sequencing problem. In this set of experiments, the performances of L–SA (Bai et al. 2007) and CF–MC are compared to the Simple Random–EMCQ and Choice Function–EMCQ hyper-heuristics. To our knowledge, this comparison has not been performed before in literature for any problem. EMCQ is a time dependent move acceptance method as presented in (14). Some initial experiments are performed to determine how many steps are taken in a unit time, so that the same framework could be respected.

As shown in Table 2, L–SA outperforms both of these EMCQ based hyper-heuristics improving their best results by approximately 82% on average for each problem instance using any termination criterion. Choice Function–EMCQ performs better than Simple Random–EMCQ in finding the best for at least nine out of thirteen problem instances regardless of the termination criterion.

The empirical cumulative distribution function for L–SA and Choice Function–EMCQ with different termination criterion is plotted in Fig. 6. {car92 I, yor83 I} is used to provide representative cases for comparing the average performances of L–SA and Choice Function–EMCQ, since a similar phenomenon is observed for the rest of the problems. The worst result provided by L–SA in fifty runs is always better than the best result provided by Choice Function–EMCQ. The Wilcoxon test for each problem instance over fifty runs confirms that

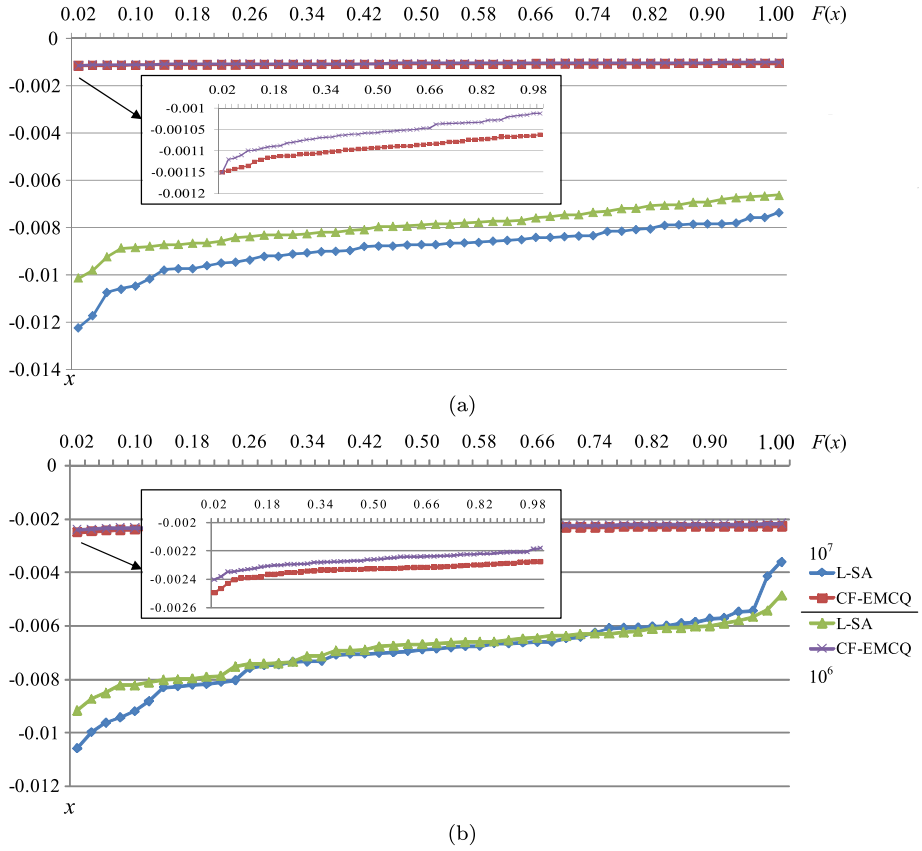

Fig. 6 Empirical cumulative distribution function of the solution quality based on the best result obtained at each run using the L–SA and CF–EMCQ hyper-heuristics with maxIteration $= 10^6$ and maxIteration $= 10^7$ for (**a**) car92 I, and (**b**) yor83 I

L–SA performs significantly better than EMCQ based hyper-heuristics within a confidence interval of 95% for all problem instances.

The performances of EMCQ based hyper-heuristics are also compared to the MC based hyper-heuristics over hecs92 I, ear83 I, tre92, lse91 and car91 I. CF–EMCQ, SR–EMCQ, CF–MC and SR–MC hyper-heuristics are ranked from 1 to 4 for each problem instance, indicating best to worst, respectively. The average ranks of CF–EMCQ, SR–EMCQ, CF–MC and SR–MC for 10^6 iterations are 3.4, 3.6, 1.40 and 1.60, respectively. CF–MC improves almost on all best solutions produced by CF–EMCQ and SR–EMCQ in any case. The %-improvements obtained for each problem instance in different settings are presented in Table 3. The results show that EMCQ performs poorly. Even the standard simulated annealing move acceptance outperforms EMCQ hyper-heuristics regardless of the heuristic selection method.

4.4 Comparison of the SA based hyper-heuristics

Different SA based hyper-heuristics are implemented using different heuristic selection methods, namely; Choice Function, Greedy and Simple Random. L–SA delivers a very poor

Table 3 %–improvement generated by CF–MC over CF–EMCQ and SR–EMCQ for each problem instance as the maximum number of iterations change from 10^6 to 10^7

	CF–MC %-improvement			
	maxIteration $= 10^6$		maxIteration $= 10^7$	
problem	CF–EMCQ	SR–EMCQ	CF–EMCQ	SR–EMCQ
car91 I	59.16	58.71	86.62	–
ear83 I	52.68	54.68	65.48	67.49
hecs92 I	83.02	83.65	85.11	85.06
lse92	64.57	65.35	75.36	76.59
tre93	76.82	75.96	89.14	88.57

Table 4 Performance comparison of SA hyper-heuristics based on rankings (1 to 4) with respect to the best solution obtained in fifty runs. Ties are taken into account while computing the ranks

	maxIteration $= 10^6$				maxIteration $= 10^7$			
problem	L–SA	CF–SA	GR–SA	SR–SA	L–SA	CF–SA	GR–SA	SR–SA
car91 I	4	1	3	2	4	1	3	2
car92 I	4	2	1	3	4	2	1	3
ear83 I	3	1	2	4	1	4	2	3
hecs92 I	4	1	3	2	4	2	3	1
kfu93	4	2	3	1	4	1	2	3
lse91	4	2	1	3	2	1	3	4
pur93 I	3	1	4	2	3	2	4	1
rye92	3	4	2	1	4	2	1	3
sta83 I	3	1	4	2	4	1.5	3	1.5
tre92	2	1	3	4	3	1	4	2
uta92 I	3	4	1	2	3	4	1	2
ute92	4	1	3	2	1	4	2	3
yor83 I	3	4	1.5	1.5	3	2	1	4
avr.	3.38	1.92	2.42	2.27	3.08	2.12	2.31	2.50
st.dev.	0.65	1.26	1.08	0.97	1.12	1.16	1.11	1.00

performance in finding the best when compared to the rest of the hyper-heuristics as illustrated in Table 4. CF–SA delivers a superior performance with an average rank of 1.92 and 2.12 over L–SA with an average rank of 3.38 and 3.08 over all problem instances when the maximum number of iterations is 10^6 and 10^7, respectively.

The average performance comparison between L–SA and CF–SA based on the Wilcoxon test and the average %-improvement of the best approach over the other one for a given problem instance are provided in Table 5. The average performance of CF–SA is better than L–SA. It is observed that even SR–SA performs better than L–SA on average. The learning mechanism loses against a random choice heuristic selection. Bai et al. (2007) claims that L–SA yields a poor performance if hill climbers are used as low level heuristics. Yet, these heuristics are not hill climbers in this study. The learning mechanism simply fails to discover the best low level constraint based heuristic to employ at a step using historical information. On the other hand, a Choice Function combined with the same move acceptance provides a better learning mechanism.

Table 5 Average performance comparison of the L–SA and CF–SA hyper-heuristics. A > B indicates that the approach A performs significantly better than B within a confidence interval of 95% based on the Wilcoxon test over fifty runs for a given problem, while A≈B indicates that A performs slightly better than B and this performance variation is not statistically significant

problem	maxIteration = 10^6 Performance	Avr. %-impr.	maxIteration = 10^7 Performance	Avr. %-impr.
car91 I	CF–SA > L–SA	26.9	CF–SA > L–SA	52.9
car92 I	CF–SA > L–SA	4.4	CF–SA > L–SA	19.2
ear83 I	CF–SA ≈ L–SA	4.9	L–SA ≈ CF–SA	–
hecs92 I	CF–SA > L–SA	29.9	CF–SA > L–SA	31.0
kfu93	CF–SA ≈ L–SA	8.6	CF–SA ≈ L–SA	9.7
lse91	CF–SA ≈ L–SA	4.3	CF–SA ≈ L–SA	8.9
pur93 I	CF–SA > L–SA	10.4	CF–SA > L–SA	2.1
rye92	L–SA ≈ CF–SA	–	CF–SA ≈ L–SA	5.6
sta83 I	CF–SA ≈ L–SA	0.1	CF–SA ≈ L–SA	0.1
tre92	CF–SA ≈ L–SA	7.1	CF–SA ≈ L–SA	25.0
uta92 I	L–SA ≈ CF–SA	–	L–SA ≈ CF–SA	–
ute92	CF–SA ≈ L–SA	5.0	L–SA ≈ CF–SA	–
yor83 I	L–SA > CF–SA	–	CF–SA > L–SA	1.3

5 Conclusion

Hyper-heuristics are emerging as simple to implement methodologies for solving difficult problems. As a sub-type, selection hyper-heuristics utilise perturbative (improvement) low level heuristics. Such a hyper-heuristic attempts to improve an initial solution iteratively by selecting the most appropriate heuristic to employ and deciding whether to accept or reject the outcome at each step. In this paper, a set of new and previously proposed Monte Carlo based selection hyper-heuristics are investigated over a set of examination timetabling benchmark data. In almost all existing hyper-heuristics, an improving move is accepted. Monte Carlo move acceptance methods also make use of this strategy. Furthermore, they allow acceptance of worsening moves based on a parametric probability distribution function.

As discussed in Özcan et al. (2008) and Bilgin et al. (2007), the choice of hyper-heuristic components and the nature of low level heuristics affect the overall performance of a hyper-heuristic. As a move acceptance approach, simulated annealing shows potential. It is observed that the use of reheating within simulated annealing might improve the performance of a hyper-heuristic even more. Simple Random and Greedy heuristic selection methods are blind strategies. They get no feedback from the search process. On the other hand, Choice Function and L heuristic selection methods dynamically get information and attempt to learn to choose the right heuristic at a given decision point during the search. Although L–SA performs better than some hyper-heuristics utilising standard simulated annealing move acceptance methods and EMCQ, it fails significantly against hyper-heuristics using Simple Random, Greedy, Choice Function heuristic selection methods instead of L. Even increasing the maximum number of iterations does not help. Choice function and L schemes are based on reinforcement learning (Kaelbling et al. 1996). Heuristics are rewarded by means of a utility function as they are successful in improving a solution or being accepted by the move acceptance. Nareyek (2003) shows that using maximal utility value for heuristic selection performs better. The Choice Function makes use of this strategy, while L does not. This

might be one of the reasons why L–SA performs worse than Choice Function–SA. Also, L rewards a heuristic whenever the resulting solution is accepted whether the move yields an improvement or not. Considering that L uses a short term memory, such a strategy might be deceptive while choosing a low level heuristic.

In order to achieve a high level of generality, a hyper-heuristic should perform consistently across different problem domains. In the context of selection hyper-heuristics based on perturbative low level heuristics, it is vital to form the right coupling between heuristic selection and move acceptance. Simulated annealing with reheating turns out to be a very promising hyper-heuristic component based on the results that are obtained from previous studies and this one. Furthermore, the work described in this paper contributes to the goal of understanding the underlying relationship between heuristic selection and move acceptance linking the heuristic design process and learning.

References

Abdullah, S., Ahmadi, S., Burke, E. K., Dror, M., & McCollum, B. (2007). A tabu-based large neighbourhood search methodology for the capacitated examination timetabling problem. *Journal of the Operational Research Society*, *58*, 1494–1502.

Alkan, A., & Özcan, E. (2003). Memetic algorithms for timetabling. In *Proc. of the congress on evolutionary computation* (Vol. 3, pp. 1796–1802).

Asmuni, H., Burke, E. K., & Garibaldi, J. M. (2005). Fuzzy multiple ordering criteria for examination timetabling. In *Lecture notes in computer science: Vol. 3616. Selected papers from the 5th international conference on the practice and theory of automated timetabling* (pp. 334–353). Berlin: Springer.

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)*, Chiang Mai, Thailand (pp. 132–141).

Azimi, Z. N. (2005). Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, *163*(2), 705–733.

Bai, R., & Kendall, G. (2005). An investigation of automated planograms using a simulated annealing based hyper-heuristics. In T. Ibaraki, K. Nonobe, & M. Yagiura (Eds.), *Operations research/computer science interface series: Vol. 32. Metaheuristics: progress as real problem solver* (pp. 87–108). Berlin: Springer.

Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., & McCollum, B. (2007). *A simulated annealing hyper-heuristic methodology for flexible decision support* (Tech. Rep. NOTTCS-TR-2007-8). School of CSiT, University of Nottingham.

Bilgin, B., Özcan, E., & Korkmaz, E. E. (2007). An experimental study on hyper-heuristics and exam timetabling. In *Lecture notes in computer science: Vol. 3867. Practice and theory of automated timetabling VI (PATAT 2006)* (pp. 394–412). Berlin: Springer.

Broder, S. (1964). Final examination scheduling. *Communications of the ACM*, *7*, 494–498.

Burke, E. K., & Newall, J. P. (1999). A multistage evolutionary algorithm for the timetable problem. *IEEE Trans Evolutionary Computation*, *3*(1), 63–74.

Burke, E. K., & Newall, J. P. (2004). Solving examination timetabling problems through adaption of heuristic orderings. *Annals of Operations Research*, *129*, 107–134.

Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, *140*(2), 266–280.

Burke, E. K., Newall, J. P., & Weare, R. F. (1996a). A memetic algorithm for university exam timetabling. In *Lecture notes in computer science: Vol. 1153. Selected papers from the first international conference on practice and theory of automated timetabling* (pp. 241–250). Berlin: Springer.

Burke, E. K., Elliman, D. G., Ford, P. H., & Weare, R. F. (1996b). Examination timetabling in British universities: a survey. In *Lecture notes in computer science: Vol. 1153. Selected papers from the first international conference on practice and theory of automated timetabling* (pp. 76–90). Berlin: Springer.

Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P., & Schulenburg, S. (2003). Hyper-heuristics: An emerging direction in modern search technology. In F. Glover & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 457–474). Norwell: Kluwer Academic.

Burke, E. K., Hyde, M. R., & Kendall, G. (2006). Evolving bin packing heuristics with genetic programming. In *Lecture notes in computer science: Vol. 4193. Proceedings of the 9th international conference on parallel problem solving from nature (PPSN 2006)*, Reykjavik, Iceland (pp. 860–869). Berlin: Springer.

Burke, E. K., Hyde, M. R., Kendall, G., & Woodward, J. (2007a). Automatic heuristic generation with genetic programming: evolving a jack-of-all-trades or a master of one. In *GECCO '07: proceedings of the 9th annual conference on genetic and evolutionary computation* (pp. 1559–1565). New York: ACM. doi:10.1145/1276958.1277273.

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

Caramia, M., Dell'Olmo, P., & Italiano, G. F. (2001). New algorithms for examination timetabling. In *Lecture notes in computer science: Vol. 1982. WAE '00: the 4th international workshop on algorithm engineering* (pp. 230–242). London: Springer.

Caramia, M., Dellolmo, P., & Italiano, G. F. (2008). Novel local search-based approaches to university examination timetabling. *INFORMS Journal on Computing*, *20*(1), 86–99.

Carter, M. W. (1986). A survey of practical applications of examination timetabling algorithms. *Operations Research Society of America*, *34*(2), 193–202.

Carter, M. W., & Laporte, G. (1996). Recent developments in practical examination timetabling. In *Lecture notes in computer science: Vol. 1153. Selected papers from the first international conference on practice and theory of automated timetabling* (pp. 373–383). Berlin: Springer.

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

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, *45*(1), 41–51.

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

Corne, D., Ross, P., & Fang, H. L. (1994). Fast practical evolutionary timetabling. In *Selected papers from AISB workshop on evolutionary computing* (pp. 250–263).

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 *Parallel problem solving from nature—PPSN IX* (pp. 392–401).

Cowling, P., Kendall, G., & Soubeiga, E. (2001a). A hyperheuristic approach to scheduling a sales summit. In *PATAT '00: selected papers from the third international conference on practice and theory of automated timetabling III* (pp. 176–190). London: Springer.

Cowling, P., Kendall, G., & Soubeiga, E. (2001b). A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the 4th metaheuristic international conference* (pp. 127–131).

Cowling, P., Kendall, G., & Soubeiga, E. (2002). Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In *Lecture notes in computer science: Vol. 4193. EvoWorkShops* (pp. 1–10). Berlin: Springer.

Dowsland, K., & Thompson, J. (2005). Ant colony optimization for the examination scheduling problem. *Journal of the Operational Research Society*, *56*(4), 426–438.

Dowsland, K. A., Soubeiga, E., & Burke, E. (2007). A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation. *European Journal of Operational Research*, *179*(3), 759–774.

Eley, M. (2006). Ant algorithms for the exam timetabling problem. In *Proc. of the 5th international conference on the practice and theory of automated timetabling* (pp. 364–382).

Ergul, A. (1996). Ga-based examination scheduling experience at middle east technical university. In *Lecture notes in computer science: Vol. 1153. Practice and theory of automated timetabling* (pp. 212–226). Berlin: Springer.

Ersoy, E., Özcan, E., & Uyar, S. (2007). Memetic algorithms and hyperhill-climbers. In *Proc. of the 3rd multidisciplinary int. conf. on scheduling: theory and applications* (*MISTA'07*) (pp. 159–166).

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

Gaspero, L. D., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. In E. K. Burke & W. Erben (Eds.), *Lecture notes in computer science: Vol. 2079. Third international conference on practice and theory of automated timetabling, PATAT2000* (pp. 104–117). Berlin: Springer.

Gogos, C., Alefragis, P., & Housos, E. (2010). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*. doi:10.1007/s10479-010-0712-3.

Goldberg, DE (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley.

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

Kendall, G., & Hussin, N. M. (2005). A tabu search hyper-heuristic approach to the examination timetabling problem at the Mara University of Technology. In *Lecture notes in computer science: Vol. 3616. Practice and theory of automated timetabling V* (pp. 270–293). Berlin: Springer.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Krasnogor, N., & Gustafson, S. (2004). A study on the use of 'self-generation' in memetic algorithms. *Natural Computing*, *3*(1), 53–76.

Leighton, F. T. (1979). A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, *84*, 489–506.

Lundy, M., & Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical Programming*, *34*, 111–124.

Marin, H. T. (1998). *Combinations of ga and csp strategies for solving the examination timetabling problem* (PhD thesis). Instituto Technologico y de Estudios Superiores de Monterrey.

McCollum, B. (2006). University timetabling: Bridging the gap between research and practice. In *Proc. of the 5th international conference on the practice and theory of automated timetabling* (pp. 15–35). Berlin: Springer.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: the second international timetabling competition. *INFORMS Journal on Computing*, *22*, 120–130.

Merlot, L. T., Boland, N., Hughes, B. D., & Stuckey, P. J. (2003). A hybrid algorithm for the examination timetabling problem. In *Lecture notes in computer science: Vol. 1153. Practice and theory of automated timetabling IV, PATAT 2002* (pp. 207–231). Berlin: Springer.

Müller, T. (2009). Itc2007 solver description: A hybrid approach. *Annals of Operations Research*, *172*(1), 429–446.

Nareyek, A. (2003). Choosing search heuristics by non-stationary reinforcement learning. In M. G. C. Resende & J. P. de Sousa (Eds.), *Metaheuristics: computer decision-making* (pp. 523–544). Norwell: Kluwer Academic, Chap. 9.

Ong, Y. S., Lim, M. H., Zhu, N., & Wong, K. W. (2006). Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, *36*(1), 141–152.

Özcan, E., & Alkan, A. (2007). A memetic algorithm for solving a timetabling problem: An incremental strategy. In: P. Baptiste, G. Kendall, A.M. Kordon & F. Sourd (Eds.), *Proc. of the 3rd multidisciplinary int. conf. on scheduling: theory and applications* (pp. 394–401).

Özcan, E., & Ersoy, E. (2005). Final exam scheduler—fes. In *Proc. of the congress on evolutionary computation* (pp. 1356–1363). New York: IEEE Press.

Özcan, E., Bilgin, B., & Korkmaz, E. E. (2006). Hill climbers and mutational heuristics in hyperheuristics. In *Lecture notes in computer science: Vol. 4193. Proceedings of the 9th international conference on parallel problem solving from nature (PPSN 2006)*, Reykjavik, Iceland (pp. 202–211). Berlin: Springer.

Özcan, E., Bilgin, B., & Korkmaz, E. E. (2008). A comprehensive survey of hyperheuristics. *Intelligent Data Analysis*, *12*(1), 3–23.

Paquete, L. F., & Fonseca, C. M. (2001). A study of examination timetabling with multiobjective evolutionary algorithms. In *Proc. of the 4th metaheuristics international conference (MIC 2001)* (pp. 149–154).

Petrovic, S., Patel, V., & Yang, Y. (2005). Examination timetabling with fuzzy constraints. In *Lecture notes in computer science: Vol. 3616. The 5th int. conf. on the practice and theory of automated timetabling* (pp. 313–333). Berlin: Springer.

Petrovic, S., Yang, Y., & Dror, M. (2007). Case-based selection of initialisation heuristics for metaheuristic examination timetabling. *Expert Systems with Applications*, *33*(3), 772–785.

Qu, R., Burke, E. K., McCollum, B., Merlot, L., & Lee, S. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*(1), 55–89.

Ross, P., Corne, D., & Fang, H. L. (1994). Improving evolutionary timetabling with delta evaluation and directed mutation. In *PPSN III: proceedings of the international conference on evolutionary computation. The third conference on parallel problem solving from nature* (pp. 556–565). London: Springer.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, *13*(2), 87–127.

Schaerf, A. & Gaspero, L.D. (2006). Measurability and reproducibility in timetabling research: State-of the-art and discussion (invited paper). In *Proc. of the 6th int. conf. on the practice and theory of automated timetabling* (pp. 53–62).

Wong, T., Cote, P., & Gely, P. (2002). Final exam timetabling: a practical approach. In *Proc. of the IEEE canadian conference on electrical and computer engineering* (Vol. 2, pp. 726–731).