Soria-Alcaraz JA, Ochoa G, Swan J, Carpio M, Puga H & Burke E (2014) Effective learning hyper-heuristics for the course timetabling problem, *European Journal of Operational Research*, 238 (1), pp. 77-86.

DOI: 10.1016/j.ejor.2014.03.046

Effective Learning Hyper-heuristics for the Course Timetabling Problem

Jorge A. Soria-Alcaraz[a,*], Gabriela Ochoa[b], Jerry Swan[b], Martin Carpio[a], Hector Puga[a], Edmund K. Burke[b]

[a] *Instituto Tecnológico de León,Doctorado en Ciencias de la Computación, División de Estudios de Posgrado e Investigación, León Guanajuato, México.*
[b] *Computing Science and Mathematics, University of Stirling, Stirling, Scotland, UK*

Abstract

Course timetabling is an important and recurring administrative activity in most educational institutions. This article combines a general modeling methodology with effective learning hyper-heuristics to solve this problem. The proposed hyper-heuristics are based on an iterated local search procedure that autonomously combines a set of move operators. Two types of learning for operator selection are contrasted: a static (offline) approach, with a clear distinction between training and execution phases; and a dynamic approach that learns on the fly. The resulting algorithms are tested over the set of real-world instances collected by the first and second International Timetabling competitions. The dynamic scheme statistically outperforms the static counterpart, and produces competitive results when compared to the state-of-the-art, even producing a new best-known solution. Importantly, our study illustrates that algorithms with increased autonomy and generality can outperform human designed problem-specific algorithms.

Keywords: Timetabling, Hyper-Heuristics, Heuristics, Metaheuristics, Combinatorial optimization

1. Introduction

The design of timetables is a widespread human activity that can be formulated as an optimization problem, and thus solved using modern search methodologies. Educational timetabling is a widely studied class of timetabling problems concerning the scheduling of meetings between students and lecturers. Several variants of educational timetabling problems have been studied in the

*corresponding author
Email addresses: soajorgea@gmail.com Tel(52)477-7040197 (Jorge A. Soria-Alcaraz), goc@cs.stir.ac.uk (Gabriela Ochoa), jsw@cs.stir.ac.uk (Jerry Swan), jmcarpio61@hotmail.com (Martin Carpio), pugahector@yahoo.com (Hector Puga), e.k.burke@stir.ac.uk (Edmund K. Burke)

literature (Adriaen et al., 2006). The *Course Timetabling Problem* is one of such variants, in which a number of events or lectures of university courses need to be scheduled over a prefixed period of time (normally a week), satisfying various constraints on rooms, time-slots and students. Many articles related to educational timetabling have been published, and automated approaches to timetabling are used in practice. A number of survey articles have appeared over the years from the early (Carter, 1986; Schaerf, 1999) to more recent articles (Burke and Petrovic, 2002; Lewis, 2008; Qu et al., 2009b). Two recent approaches include an adaptive linear combination of heuristics (Rahman et al., 2014) and a two-stages decomposition of an integer programming model applied to a practical case (Sørensen and Dahms, 2014). The state-of-the-art approaches for course timetabling are further discussed in Section 4.3.

Automating the design of timetables is a complex task. It requires a detailed and difficult to elicit knowledge of the problem and the particular instance to be solved. A recent trend in search and optimization, *hyper-heuristics*, aims at reducing the role of the human expert in the process of designing computational search methodologies, and thus raise the level of generality in which these methodologies operate (Burke et al., 2003, 2010). A survey of the state of the art in hyper-heuristics has been recently published (Burke et al., 2013). Hyper-heuristic approaches have been applied to educational timetabling with encouraging results (Qu et al., 2009a; Burke et al., 2007; Soria-Alcaraz Jorge et al., 2010). An automated approach to course timetabling also requires the ability to model different classes of problems with a wide range of characteristics and constraints. The modeling methodology proposed in (Soria-Alcaraz Jorge et al., 2013b,a) (*Methodology of Design*) fills this requirement by providing a generic representation applicable to real-world instances.

An important aspect of hyper-heuristics is *generality* and how to define it. As discussed in (Misir et al., 2013), generality can be across various problem domains (Ochoa et al., 2012a,b) or across various heuristic sets (Chakhlevitch and Cowling, 2005; Misir et al., 2010). Generality has also been related to the ability to solve several variants of the same problem, which is achieved by designing problem models with increased generality. This is the case of the excellent approaches dealing with vehicle routing (Pisinger and Ropke, 2007) and nurse rostering (Burke and Curtois, 2014), respectively. This latter understanding of generality is the most relevant to our automated approach to course timetabling. We consider a single problem and a single heuristic set. Indeed a pre-processing stage is suggested to select the most adequate members of the heuristic set. However, our approach is general in that different types of course timetabling policies and constraints can be modeled. This is an important feature in real-world timetabling, as it is common that whenever a change emerges in institutional policies (adding, removing or changing current timetabling constraints) it is necessary to change the previously used algorithm to handle these new situation. The proposed general modeling methodology is then combined with an adaptive search algorithm incorporating learning mechanisms and effective move operators to produce a state-of-the art approach.

This article implements an iterated local search hyper-heuristic framework

that combines several move operators for the course timetabling problem. The autonomous design of algorithms requires the incorporation of machine learning mechanisms. The article contrasts two types of learning: static (offline) learning, in which there is a clear distinction between a training phase and an execution phase; and dynamic (online) learning, which takes place while the algorithm is solving a given instance. The resulting algorithms are tested over the set of publicly available real-world instances collected from the first and second International Timetabling competitions (McCollum et al., 2010).

The next section formulates the course timetabling problem and describes the modeling methodology used. Section 3 describes the proposed iterated local search hyper-heuristic framework, including the learning mechanisms and the move operators considered. The longest section in the article, Section 4, describes the group of empirical studies conducted and analyzes the results. Finally, Section 5 summarizes the main findings and suggests directions for future work.

2. The Course Timetabling Problem

The course timetabling problem can be formulated as a constraint satisfaction problem in which the variables are events. The problem may be concisely defined (Conant-Pablos et al., 2009), in terms of a set of events (courses or subjects) $E = \{e_1, e_2, \ldots, e_n\}$, a set of time-periods $T = \{t_1, t_2, \ldots, t_s\}$, a set of places (classrooms) $P = \{p_1, p_2, \ldots, p_m\}$, and a set of agents (students registered in the courses) $A = \{a_1, a_2, \ldots, a_o\}$. An assignment is then given by the quadruple $(e \in E, t \in T, p \in P, S \subseteq A)$, and a solution to the problem is a complete set of n assignments (one for each event) that satisfies the set of hard constraints.

The modelling methodology proposed in (Soria-Alcaraz Jorge et al., 2013a,b) (*methodology of design*) is used here to represent the course timetabling instances and their constraints. This methodology allows different types of course timetabling policies and constraints to be modelled by converting all time and space constraints into a single constraint type: *student conflicts*.

The ability to model across different institutions and formulations is achieved by translating the original timetable instance data into a set of generic data structures which are later used by the hyper-heuristic. Specifically, the information is integrated into three data structures: i) *subject-subject matrix (MMA)* , ii) *list of possible timeslots (LPH)* and iii) *list of possible classrooms* (LPA). As described in (Soria-Alcaraz Jorge et al., 2013a,b), an important advantage of this methodology is that the generic structures facilitate the construction of feasible solutions. The subject-subject matrix contains the number of students in conflict for a given pair of subjects, i.e. the number of students who are enrolled in a given pair of subjects. The LPH list contains the allowed timeslots for the corresponding subject. The LPA list contains the classrooms available to be assigned to each subject without conflict.

These structures are then used to construct timetables. An important advantage of this methodology is that the generic structures facilitate the construction

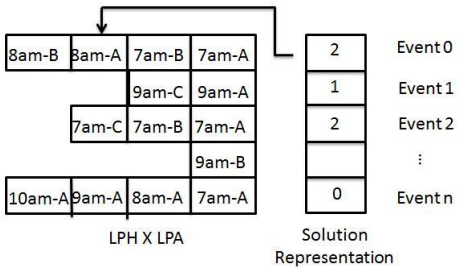

Figure 1: Representation or encoding of candidate timetable solutions

of feasible solutions. Specifically, the Cartesian product of lists LPH (timeslots) and LPA (classrooms) is constructed as shown in Figure 1. A timetable is encoded as a vector of length equal to the number of subjects, in which positions indicate subjects. The integer values in the vector are indices representing a pair (LPH_i, LPA_i), where LPH_i is a valid timeslot for the subject i and LPA_i is a valid classroom for the subject i. This vector represents a complete timetable assignment. In this formulation, the search space is then given by the Cartesian product $LPH_i \times LPA_i$, and the objective is to reduce the total number of student conflicts. Table 1 gives information about the problem instances considered.

Table 1: Course timetabling instances.

Family	Source	Number
ITC-2007	`http://www.cs.qub.ac.uk/itc2007/`	24
ITC-2002	`http://www.idsia.ch/Files/ttcomp2002/`	20

3. The Iterated Local Search Hyper-Heuristic

Hyper-heuristics search the space of heuristics rather than that of solutions, and use limited problem specific information to control the search process. The problem specific information is encapsulated into the problem model and a pool of low-level heuristics or search operators.

The proposed hyper-heuristic strategy can be seen as an adaptive version of the iterated local search strategy combining several move operators. Iterated local search is a relatively simple yet powerful strategy. It operates by iteratively alternating between applying a move operator to the incumbent solution (perturbation stage) and restarting local search from the perturbed solution (improvement stage). This search principle has been rediscovered multiple times, within different research communities and with different names (Battiti et al., 2007). The term *iterated local search* (ILS) was proposed in (Lourenço et al., 2003).

A number of adaptive variants of multi-neighborhood iterated local search have been recently proposed (Ochoa et al., 2012b; Walker et al., 2012) with encouraging results in other problem domains. If several options are available for conducting perturbation and improvement, a mechanism needs to be provided to choose between them. The idea is to use online learning to adaptively select the operators either at the perturbation stage or the improvement stage; or both. These approaches inspired the algorithm implemented in this article, which can be seen in Algorithm 1 and 2. In this implementation, the perturbation stage (step 4 in Algorithm 1) applies a single fixed move operator to the incumbent solution. This move operator (Simple Random Perturbation - SRP) simply selects uniformly at random a single variable and substitutes it for another variable in the range selected uniformly at random.

Online learning is then applied to the improvement stage (Algorithm 2), in which a low-level heuristic from the pool is selected and applied to the incumbent solution (steps 3 and 4). This operator selection step uses learned probabilities to conduct the selection. The process of learning the operator probabilities is conducted using online (dynamic) and offline (static) schemes as detailed below. Throughout this article we use the terms 'operator', 'heuristic' and 'low-level heuristic' interchangeably.

The next subsection describes the mechanisms for adaptively operator selection. Section 3.2, describes how to statically tune operator selection probabilities. Finally, section 3.3 lists and describes the set of heuristics (operators) considered in our implementation.

Algorithm 1 High Level Iterated Local Search (ILS)

1: $s_0 = GenerateInitialSolution$
2: $s^* = ImprovementStage(s_0)$
3: **while** $!StopCriteria()$ **do**
4: $\quad s' = SimpleRandomPerturbation(s^*)$
5: $\quad s^{*'} = ImprovementStage(s')$
6: $\quad$ **if** $f(s^{*'}) < f(s^*)$ **then**
7: $\quad\quad s^* = s^{*'}$
8: $\quad$ **end if**
9: **end while**
10: **return** s^*

Algorithm 2 Improvement Stage

```
ls ← IncumbentSolution
while !LocalStopCriteria() do
    h_i = SelectHeuristic()
    ls* = apply(h_i, ls)
    if f(ls*) < f(ls) then
        ls = ls*'
    end if
end while
return ls
```

3.1. Adaptive operator selection

An adaptive operator selection scheme consists of two components: (i) a *credit assignment* mechanism, which associates a reward with each operator, modeling its predicted utility; and (ii) a *selection rule*, which determines the operator to be used at each time step, as a function of reward. We detail these below.

3.1.1. Credit assignment

We implemented the *extreme value* credit assignment, which is based on the principle that large (but possibly infrequent) improvements in the objective score are likely to be more effective than small frequent improvements (Fialho et al., 2008). It rewards operators which have had a recent large positive impact on the objective score, while consistent operators yielding only small improvements receive less reward. Rewards are updated as follows, when a heuristic h is selected, it is applied to the current solution. The fitness of this new solution is computed and the change in fitness is added to a FIFO list of size W. This list is unique and common for all operators. Thereafter, the operator reward is updated to the maximal fitness improvement in the list. More formally, if t be the current step and $\delta(t)$ the fitness improvement observed at time t, then the expected reward for heuristic h is computed as follows (Equation 1):

$$\hat{r}_t = argmax\{\delta(t_i), i = 1 \dots W\} \tag{1}$$

As seen in equation 1), the extreme value mechanism requires a regulatory integer parameter W, the window size. If W is too small, the range of information on offer is narrowed, meaning that useful operators are missed. If it is too large, information considered may be from many iterations ago and hence no longer be relevant.

3.1.2. Selection rule

Operator selection rules typically associate probabilities with operators via proportional selection. Let K denote the number of operators (low-level heuristics). The selection mechanisms maintain a probability vector $(p_i, t)_{i=1,\dots,K}$, and an estimate of the current operator credit denoted as $\hat{q}_{i,t}$. At each iteration time t:

- Operator i is selected with probability $p_{i,t}$, according to a roulette-wheel selection scheme.
- The selected operator is applied, and a credit r_t is computed using the credit assignment mechanism (extreme-value in our implementation).
- The quality estimate $\hat{q}_{i,t}$ of the selected operator is updated according to the reward r_t, using an additive relaxation mechanism with learning rate $\alpha(0 < \alpha \leq 1)$. The learning rate controls the memory of the quality estimate. Specifically, the memory span decreases with increasing α, as indicated by Equation 2.

$$\hat{q}_{i,t+1} = (1 - \alpha) \times \hat{q}_{i,t} + \alpha \times r_t \tag{2}$$

Operator selection probabilities are calculated from the operator quality estimates following a selection rule. Two commonly rules, namely *Probability Matching* (PM) (Goldberg, 1990) and *Adaptive Pursuit* (AP) (Thierens, 2005), were implemented and tested in this article.

Probability Matching corresponds to the standard *roulette wheel* selection. The goal is to make $p_{i,t}$ proportional to $\hat{q}_{i,t}$. An operator that performs very badly during a long period of the search will have its quality estimate decreased to a very low value, or even zero. To avoid such operators being completely ignored, the selection rules normally assign a minimal selection probability $p_{min} > 0$. Equation 3 describes the PM rule:

$$p_{i,t+1} = p_{min} + (1 - K * p_{min}) \frac{\hat{q}_{i,t+1}}{\sum_{j=1}^{K} \hat{q}_{j,t+1}} \tag{3}$$

In contrast, *Adaptive Pursuit* follows a winner-take-all strategy, selecting at each step the operator i_t^* with maximal reward, and increasing its selection probability accordingly. Equations 4, 5 and 6 describes this mechanism:

$$i^* = argmax\{\hat{q}_{i,t}, i = 1 \dots K\} \tag{4}$$

$$p_{i^*,t+1} = p_{i^*,t} + \beta(1 - (K - 1)p_{min} - p_{i^*,t}), (\beta > 0) \tag{5}$$

$$p_{i,t+1} = p_{i,t} + \beta(p_{min} - p_{i,t}), \text{for } i \neq i^* \tag{6}$$

As discussed above, adaptive operator selection mechanisms introduce new parameter values. The values used in our hyper-heuristic implementation are reported in Section 4.2, specifically, in Table 4. They were selected after some preliminary experiments.

3.2. Automated algorithm configuration (offline tuning)

The previous section discussed mechanisms for dynamically adapting the operators selection probabilities. These probabilities can alternatively be considered as fixed (static) parameters of the ILS hyper-heuristic algorithm that can then be automatically tuned. Operators are then selected by a roulette-wheel mechanism based on these statically-tuned probabilities. Several frameworks

have been proposed in the literature for automated parameter tuning and algorithm configuration (Nannen and Eiben, 2006; Birattari, 2009; Hutter et al., 2009). We use here *ParamILS*, a framework for automated algorithm configuration achieved via a local search in the configuration space. The key idea behind ParamILS is to combine a stochastic local search algorithm (iterated local search) with mechanisms for exploiting specific properties of algorithm configuration. The search process starts from a given configuration (which is generally the target algorithm's default configuration) as well as r additional configurations chosen uniformly at random from the given discrete ranges of the configuration parameters. These $r + 1$ initial configurations are evaluated, and the best performing is selected as the starting point of the ILS. To evaluate a configuration, the idea is to perform a fixed number of runs of the target algorithm with the given configuration on the set of training instances. The process proceeds with the iterated local search strategy, using the one-exchange neighborhood (i.e. inducing an arbitrary change in a single target algorithm parameter) in the improvement stage, and a number of steps s of the same neighborhood as the perturbation stage. Small values of s (i.e. $s = 2$) have been found to be sufficient for obtaining good performance of the overall configuration procedure (Hoos, 2012). In Hutter et al. (Hutter et al., 2009), extensive evidence is presented that ParamILS can find substantially improved parameter configurations of complex and highly optimized algorithms.

Section 4.2 (specifically, Table 5) reports the values obtained after tuning operator selection probabilities, using ParamILS, in our hyper-heuristic implementation.

3.3. The Operator Pool

A pool of 9 low-level heuristics were implemented. These range from simple randomized exchange or swap neighborhoods to greedy and more informed procedures. We proposed two novel operators, *Statistical Dynamic* and *Double Dynamic* perturbations, which consider a probability distribution based on the frequency of variables selection.

1. **Simple Random Perturbation (SRP):** uniformly at random chooses a variable i and changes its value for another one inside its feasible domain $LPH_i \times LPA_i$ (selected uniformly at random).
2. **Best Single Perturbation (BSP):** chooses a variable following a sequential order (according to the Cartesian product $LPH \times LPA$) and changes its value to that producing the minimum conflict. A record is kept of the last selected variable, so the order is continued when the heuristic is called again.
3. **Statistical Dynamic Perturbation (SDP):** chooses a variable following a probability distribution based on the frequency of variable selection in the last k iterations. Variables with lower frequency will have a higher probability of being selected. Once selected, the value is randomly changed. This heuristic is an original contribution of this article. Algorithm 3 describes the pseudo-code.

4. **Double Dynamic Perturbation (DDP):** similar to heuristic SDP, in that it receives an input solution and selects a new variable with a probability inversely proportional to its frequency of selection in the last k iterations. It differs from SDP in that it internally maintains an additional solution (which is copy of the first initialized solution) and makes random changes to it following the same distribution. The best of the two modified solutions is returned. This heuristic is an original contribution of this article. Algorithm 4 describes the pseudo-code.
5. **Swap (SWP):** selects two variables uniformly at random and interchanges their values if possible. Otherwise leaves the solution unchanged.
6. **Two Points Perturbation (2PP):** selects uniformly at random two indices in the integer string representation and modifies all variables between the indices with randomly selected feasible values. This yields a strong perturbation.
7. **Move to Less Conflict (MLC):** locates the variable producing the most conflicts and changes its value to the that causing the minimum possible conflict. In other words, it applies the *Best Fit* principle.
8. **Burke-Abdullah (BA):** hybrid heuristic that chooses a variable by means of either Fail-First (FF) or Brelaz Heuristic (BZ)(Gent et al., 1996) and then changes its value according the best possible solution obtained by the following 4 algorithms: *Sequential selection, Least Constrained Value, Randomly or Minimum Conflict.* (Abdullah et al., 2007)
9. **Conant-Pablos (LSA):** hybrid heuristic that randomly selects a variable with hard constraint conflicts and changes it by a feasible value selected using either the *Minimum Constraint or Least Constrained Value* (Gent et al., 1996) heuristics (Conant-Pablos et al., 2009).

Algorithm 3 Statistical Dynamic Perturbation (SDP)

Require: $Solution, K, HistoryList(K), Frequency[Solution.NumOfVars]$
1: $Solution_{new} = Copy(Solution)$
2: $UpdateFrequency(HistoryList, Frequency)$
3: $Selection_t = SelectVariableWithDistribution(Frequency)$
4: **if** $HistoryList.getRecorderSteps() >= K$ **then**
5: $\quad HistoryList.remove(Step_0)$
6: **end if**
7: $HistoryList.add(Selection_t)$
8: $AssignRandomlyValue(Selection_t, Solution_{new})$
9: $t = t + 1$
10: $Return(Solution_{new})$

Algorithm 4 Double Dynamic Perturbation (DDP)

Require: $InnerSolution, Solution, K, HistoryList(K), Frequency[Solution.NumOfVars]$
1: **if** $t == 0$ **then**
2: $\quad InnerSolution = Copy(Solution)$
3: **end if**
4: $Solution_{new} = Copy(Solution)$
5: $UpdateFrequency(HistoryList, Frequency)$
6: $Selection_t = SelectVariableWithDistribution(Frequency)$
7: **if** $HistoryList.getRecorderSteps() >= K$ **then**
8: $\quad HistoryList.remove(Step_0)$
9: **end if**
10: $HistoryList.add(Selection_t)$
11: $AssignRandomlyValue(Selection_t, Solution_{new})$
12: $AssignRandomlyValue(Selection_t, InnerSolution)$
13: $t = t + 1$
14: **if** $Evaluation(InnerSolution) <= Evaluation(Solution_{new})$ **then**
15: $\quad Solution_{new} = Copy(InnerSolution)$
16: **end if**
17: $Return(Solution_{new})$

The choice of heuristics for the pool is somewhat arbitrary and reflects our knowledge of the representation and the problem. Since there is no fixed procedure in the literature for guiding the selection of the best set of heuristics for a given representation and problem, choices are generally based on empirical evidence. In order to refine our initial choice of operators, we propose a single test procedure described in detail in Section 4.1.

4. Experiments and Results

Two sets of well-known real-world instances are considered for our experiments, namely, those of the first and second International Timetabling competitions: *ITC-2002* and *ITC-2007* (see Table 1). With the objective to measure how good our approach performs with different problem domains in a similar way a described in Misir et al. (2013).

This section gives an overview of the empirical studies conducted. It is structured in four subsections as follows. Section 4.1 describes a methodology for selecting an effective subset of low-level heuristics from the set described in Section 3.3. Section 4.2 compares the alternative hyper-heuristic learning mechanisms discussed in Sections 3.1 and 3.2. Section 4.3 compares the best performing learning hyper-heuristic with state-of-the-art approaches on the *ITC-2007* benchmark instances. Finally, Section 4.4, presents a study of the learned probabilities and frequency of operator selection of the best-performing learning hyper-heuristic.

Parametric statistical tests are used in the literature to contrast the performance of competing algorithms. However, they are based on assumptions (i.e. independence, normality, and homoscedasticity) that are likely violated when considering stochastic search algorithms (Derrac et al., 2011). Nonparametric statistical procedures overcome this limitation and can be used for comparing

this type of algorithm. In this article we used CONTROLTEST, a tool for nonparametric comparison between algorithms (Derrac et al., 2011)[1]. Specifically, three non-parametric tests were conducted: Friedman, Aligned Friedman, and Quade. These tests report the *average* rank achieved by each algorithm across a set of independent experiments. The Friedman test uses the arithmetic mean. The Alignment Friedman uses a value of location computed as the average performance achieved by all algorithms in each problem. The Quade test considers that some problems might be more difficult than others. Therefore, the average rankings computed on each problem could be scaled depending on the differences observed in the algorithms performances, obtaining, as a result, a weighted ranking analysis of the results. Since all these tests consider ranks, the lower the reported average values the better the performance. In order to assess the statistical significance of the results, the *p-value* for each test is computed, which provides information about whether a statistical hypothesis test is significant or not. In our study the null hypothesis H_0 represents no significant differences between algorithms. The *p-values* also indicate how significant the results are: the smaller the *p-value* the stronger the evidence against H_0.

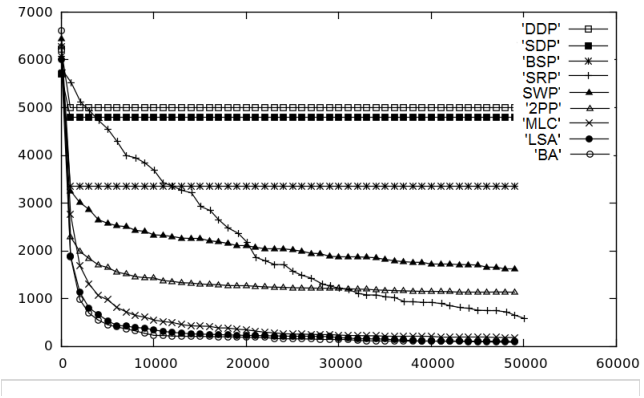

Figure 2: Average best-so-far objective value over number of evaluations for single heuristic hill-climber on a selected instance (*ITC-2007-16*). The horizontal axis measures function evaluations, while the vertical axis measures the objective function value

4.1. Choosing the heuristic pool

This section proposes an empirical methodology to select the most effective operators from the set of available heuristics described in Section 3.3. The

[1]The ControlTest package is available at `http://sci2s.ugr.es/sicidm/`

idea is to implement a simple iterative improvement procedure (hill-climbing) based on each single heuristic. Specifically, Algorithm 2 is run using each of the 9 heuristics described in section 3.3, separately. For each operator, 50 independent experiments were executed with 50,000 function evaluations on all the test instances (see Section 4).

In order to assess the statistical significance of the results, Table 2 reports the average ranks computed through the Friedman, Aligned Friedman and Quade tests on the *ITC-2007* instances (similar results were found for the *ITC-2002* family). For the three tests, the smaller the average rank the better the hill-climber and by assumption, the better the move operator. As can be seen in Table 2, the order between the heuristics is the same for the three tests. For Friedman and Quade tests the average ranks go from 1 (best case) to k (where k represents the number of heuristics to compare). For Aligned Friedman the reported average ranks go from 1 (best case) to $k \times n$,where k is the number of heuristics and n the number of test instances. For the experiments reported in Table 2, the values are $k = 9$ (the number of heuristics) and $n = 24$ (the number of instances from *ITC-2007* track2). The evidence supports that there are performance differences among the heuristics. We decided to select the top 5 out of the 9 heuristics according to these statistical ranks. Namely, (i) *Double Dynamic Perturbation* (**DDP**), (ii) *Statistical Dynamic Perturbation* (**SDP**), (iii) *Best Simple Perturbation* (**BSP**), (iv) *Simple Random Perturbation* (**SRP**), and (v) *Swap* (**SWP**). These are the heuristics that produced a similar or improved performance than the simplest *Simple Random Perturbation* (SRP) heuristic, that we have adopted as our decision point.

The *p-values* (3.85E-11, 5.32281E-4, and 1.24E-14, respectively) computed using *SRP* as control method and through the statistics of the three tests strongly suggest the existence of significant differences among the algorithms considered, i.e. the null hypothesis, H_0 suggesting no difference between algorithms is strongly rejected.

Table 2: Average statistical rankings of the single heuristic hill-climbers on the *ITC-2007* benchmark instances.

Heuristic	Friedman	A. Friedman	Quade
DDP	**3.41**	**74.10**	**2.74**
SDP	3.70	78.33	3.67
BSP	4.08	83.45	3.51
SRP	4.12	85.85	4.14
SWP	4.33	92.87	4.41
2PP	5.54	124.5	4.94
MLC	6.21	151.39	5.64
LSA	7.48	197.83	7.12
BA	8.87	202.47	8.79

Figure 2 illustrates the convergence behavior of the hill-climbers using the 9 single heuristics on the *ITC-2007* benchmark. Convergence behaviors were

similar on all instances, and suggest that the bottom heuristics (specially the bottom 3) are not specially suited for the studied problem.

We conducted a set of experiments in order to statistically support the advantage of using the reduced set of 5 heuristics against considering the whole pool of 9 heuristics. The experiments compare the hyper-heuristic variants discussed below (ie *AdapExAP*, *AdapExPM*, and StaticTuned in Table 3), using the whole set of nine heuristics against using the top ranking five. Each hyper-heuristic variant was executed over 50,000 fitness functions on the *ITC-2002* and *ITC-2007* instances. The results convincingly show that hyper-heuristics with the reduced heuristic pool outperformed those using the whole heuristic set. Statistical significance was assessed by means of the Wilcoxon signed-rank test for pair-wise comparisons (5 heuristics vs. 9 heuristics). The *p-values* obtained between *DDP* and *BA* were: 3.95E-3 for *AdapExAP*, 3.45E-3 for *AdapExPM*, and 0.0277 for *StaticTuned*.

Table 3: Name and the description of the hyper-heuristic variants implemented.

Name	Description
AdapExAP	Adaptive probabilities, extreme credit + adaptive pursuit
AdapExPM	Adaptive probabilities, extreme credit + probability matching
StaticTuned	Static probabilities, tuned using ParamILS
StaticUnif	Static probabilities, uniform (equal) for all operators

Table 4: Parameters used by the adaptive operator selection mechanisms.

Adaptive rule	W	α	β	p_{min}
AdapExAP	50	0.7	0.3	0.1
AdapExPM	40	0.75	0.4	0.1

4.2. Contrasting hyper-heuristic learning mechanisms

Four hyper-heuristic variants were implemented following the algorithmic template described in Section 3 (ie. Algorithms 1 and 2). The four variants differ in the mechanism for selecting an operator or low-level heuristic in the improvement stage (Algorithm 2, line 3, *SelectHeuristic()*). Table 3 summarizes these variants indicating the name given to them in the rest of the article.

The adaptive operator selection schemes are described in detail in Section 3.1. They combine a credit assignment mechanism with an operator selection rule, both of which involve parameters that need to be set. A preliminary set of experiments suggested the values shown in Table 4 for the combinations of extreme value credit assignment (Ex) and adaptive pursuit (AP)/probability matching (PM) rules, respectively. These values were selected from the range [1,100] for W and (0,1) for α, β and p_{min} parameters.

ParamILS (see Section 3.2) was used to automatically tune the operator probabilities (of the 5 selected top operators) as fixed parameters. This methodology requires selecting a set of training instances. We used the whole *ITC-2007*

family in the training phase and and the algorithm was run for 100,000 fitness function evaluation per instance. The probability distribution found by ParamILS can be seen on table 5, which gives the same probability to 3 of the operators, a higher probability to SRP and a low probability to SWP.

Table 5: Probability distribution resulted from ParamILS training

Heuristic	DDP	SDP	BSP	SRP	SWP
Probability	0.204	0.204	0.204	0.227	0.159

The simplest algorithm in this experiment *StaticUnif*, does not involve any form of learning. Instead, it assigns fixed and equal probabilities to the 5 selected operators (i.e. 0.2 each), which is equivalent to selecting operators uniformly at random at each iteration. This algorithm is used as a control to contrast against the learning mechanisms.

In order to compare the alternative hyper-heuristic variants, Table 6 reports the average ranks computed through the Friedman, Aligned Friedman and Quade tests on the whole set of instances. The ranks are ordered according to the Quade values. In this case Friedman and Quade tests reports average ranks from 1 (best case) to $k = 4$ algorithms, and Aligned Friedman average ranks goes from 1 (best case) to $k \times n$, where $k = 4$ algorithms and $n = 24$ instances. Notice that the best-performing variant is consistently *AdapExAP*, the second and third performing variants are closer and their ranks swap with the statistical test. The worst-performing approach is consistently *StaticUnif*.

Table 6: Average rankings of the algorithms

Algorithm	Friedman	Aligned Friedman	QUADE
AdapExAP	**1.270**	**20.708**	**1.351**
StaticTuned	2.749	49.0	2.373
AdapExPM	2.354	52.6041	2.531
StaticUnif	3.6249	71.687	3.7433

Table 7: p-values (AdapExAP is the control method)

Algorithm	Friedman	Aligned Friedman	QUADE
StaticTuned	0.0836	0.0785	0.0589
AdapExPM	0.002	1.3E-4	3.4E-6
StaticUnif	6.02E-9	2.48E-4	1.6E-13

The *p-values* on table 7 computed through the statistics of the three tests strongly suggest the existence of significant differences among the algorithms considered, i.e. the null hypothesis, H_0 suggesting no difference between algorithms is strongly rejected.

Table 8 reports the *Contrast Estimation* based on medians (Doksum, 1967; García et al., 2010) for the comparative study. This statistical technique esti-

mates the performance difference between all pairs of algorithms. It assumes that the expected differences between algorithms are the same across problems. The magnitudes in the table reflect the performance differences between the pairs of competing algorithms (Derrac et al., 2011). As can be seen from Table 8, *AdapExAP* produced the largest and positive performance differences, confirming it as the best performing algorithm. The differences in performance between the 2^{nd} and 3^{rd} performing algorithm (*StaticTuned* and *AdapExPM*) are very small, suggesting that a dynamic approach for adapting the operator probabilities needs to be adequate to be effective. Finally, the inferior performance of *StaticUnif*, confirms that learning operator selection probabilities makes the algorithm not only more robust, but also more effective.

Table 8: Contrast Estimation of the four hyper-heuristic variants.

	AdapExAP	*StaticTuned*	*AdapExPM*	*StaticUnif*
AdapExAP	0.00	28.75	30.63	51.63
StaticTuned	-28.75	0.000	1.875	22.88
AdapExPM	-30.63	-1.875	0.000	21.00
StaticUnif	-51.63	-22.88	-21.00	0.0000

4.3. Comparison with state-of-the-art approaches

This section compares the performance of the most effective learning hyper-heuristic found in the previous section (viz. *AdapExAP*) with state-of-the-art approaches for post-enrollment course timetabling. By 'state-of-the-art' we mean not only the winners (specifically, the top 2 entries) of the most recent course timetabling competition *ITC-2007*, track 2 (post-enrollment) but also the most recent and effective published algorithms for solving these instances.

In order to perform a fair comparison, we followed the rules and instances of the Second International Timetabling Competition *ITC-2007* (McCollum et al., 2010), Track 2 (see Table 1). Competitors were required to obtain valid solutions (all hard constraints are satisfied) but there may be unplaced events (soft constraints). For calculating the allotted running time, the benchmark program supplied by *ITC-2007* was used. The program provides a measure for how long an algorithm can be run on a particular machine on the competition problem instances. This running time is generally between 300 and 600 seconds (per run, per instance) on a modern PC. The computer used in our experiments was an Intel core 2 CPU 6700, 2.66GHz x 2 with 2Gb of Ram and Linux Ubuntu 12.04 32-bit with LXDE using OpenJDK7.

The algorithms included in the comparison are described below:

Atsuna. The second place in the 2007 competition formulates the timetabling instances as constraint satisfaction problems, and then uses a general purpose constraint solver to find solutions. In particular, they used the solver proposed in (Nonobe and Ibaraki, 2001), which adopts a hybrid meta-heuristic algorithm combining tabu search and iterated local search, and handles weighted constraints.

Cambazard. The winner of the 2007 competition is a multi-stage local search algorithm considering several neighborhoods, and involving aspects of tabu search and simulated annealing at different stages.

Ceschia et al. (2012). Propose a single-step metaheuristic approach based on simulated annealing, working on a neighborhood composed of moves that reschedule one event or swap two events. The solver is able to deal with all the different variants of the course timetabling problem proposed in the literature, and provides new best-know solutions in many instances.

Lewis (2012). Proposes a 3-stage local search algorithm, in which a constructive phase is followed by two separate simulated annealing phases. The algorithm behavior depends on the allotted running time, as several parameters controlling the intensity of search, are calculated according to the computation time available. The algorithm obtains good results on the *ITC-2007* instances, but it is not superior to the top entries, and fails to produce new best-known solutions.

Jat and Yang (2011). Use a hybrid population-based algorithm of 2 phases. In the first phase, a guided search genetic algorithm is used to globally find good, which are then improved in a second phase using tabu search. At the time of its publication, this approach obtained competitive results against the 2007 competition entries also providing several best-known solutions.

Table 9 shows the best results (out of 10 runs, as this was the experimental setting used in the competition) of the algorithms described above when solving the 24 *ITC-2007* instances. The comparison is conducted using the 2007 competition rules and corresponding running time. Since all solutions are feasible, the values in the table correspond to the soft constraint violations, i.e. have zero hard constraint violation (except those marked with an x in Table 9). Values are taken from *ITC-2007* website (see Table 1) for the competition entries (Atsuna and Cambazard), and from the respective publications for recent state-of-the-art approaches.

The best-performing learning hyper-heuristic, *AdapExAP*, produces competitive results against the state-of-the-art, finding the best possible solution (0 soft constraint violations) in 6 out of 24 instances, and producing a new best-known solution (150) for instance *ITC-2007-20*. When compared with the 2007 competition entries, *AdapExAP* improved the best solution on 3 of the instances, namely,*ITC-2007-16*, *ITC-2007-20* and *ITC-2007-24*.

4.4. Study of learned operator probabilities and frequency of use

This section explores both the values of the learned probabilities, and the frequency of usage of the 5 operators (low-level heuristics) in the hyper-heuristic pool. The goal is to gain a deeper understanding of the relative impact and dynamic behavior of the different heuristics in the pool while solving different course timetabling instances. In particular, we selected the best-performing learning hyper-heuristic *AdapExAP*, and plotted the values of the learned probabilities across the run. Similarly, we visualize the frequency of selection of

Table 9: Comparison against state-of-the-art approaches on the 24 *ITC-2007* instances. Values indicate the best soft constraint (s) results (out of 10 runs), in all cases (except those marked by an x), solutions are feasible, i.e. the hard constraints are 0. Average and Standard deviation results are also reported in brackets for the approach proposed in this article *AdapExAP* in the form of ($\bar{s}_{\sigma}$)

ITC-2007	*Atsuna*	*Cambazard*	*Ceschia*	*Lewis*	*Jat & Yang*	*AdapExAP*
1	61	571	**59**	1166	501	650 ($780.45_{148.5}$)
2	547	993	**0**	1665	342	470 ($960.7_{270.4}$)
3	382	164	**148**	251	3770	290 ($337_{88.7}$)
4	529	310	**25**	424	234	600 ($815_{42.6}$)
5	5	5	**0**	47	**0**	35 ($39.16_{9.3}$)
6	**0**	**0**	**0**	412	**0**	20 ($29.4_{7.3}$)
7	**0**	6	**0**	6	**0**	30 ($33.74_{2.1}$)
8	**0**	**0**	**0**	65	**0**	**0** (0_{0})
9	**0**	1560	**0**	1819	989	630 ($861.1_{127.4}$)
10	**0**	2163	3	2091	499	2349 ($2458.2_{185.2}$)
11	548	**178**	142	288	246	350 ($405.7_{57.3}$)
12	869	**146**	267	474	172	480 ($506.4_{27.4}$)
13	**0**	**0**	1	298	**0**	46 ($77.37_{49.2}$)
14	**0**	1	**0**	127	**0**	80 ($108.3_{33.5}$)
15	379	**0**	**0**	108	**0**	**0** ($5.75_{9.4}$)
16	191	2	**0**	138	**0**	**0** ($2.22_{4.1}$)
17	1	**0**	**0**	**0**	**0**	**0** (0_{0})
18	**0**	**0**	**0**	25	**0**	20 (25.16_{6})
19	x	1824	**0**	2146	84	360 ($404.51_{39.1}$)
20	1215	445	543	625	297	**150**($177.12_{37.1}$)
21	**0**	**0**	5	308	0	**0** ($3.78_{5.7}$)
22	**0**	29	5	x	1142	33 ($45.71_{12.7}$)
23	438	**238**	1292	3101	963	1007 ($1378.45_{319.4}$)
24	720	21	**0**	841	274	**0** ($45.88_{60.0}$)

the heuristics, which is based on the learned probabilities, but induces some randomness since it uses the adaptive pursuit rule.

Figures 3 and 4 show the hyper-heuristic learned probabilities and frequency of use of the 5 heuristics in the pool, on two selected instances, *ITC2007-9* and *ITC2007-13*, respectively. The curves show the values taken at intervals as indicated in the horizontal axis, with a total running time following the 2007 competition benchmark. It is interesting to note that each instance has a different operator profile. There is no best operator for a given instance, instead operators cooperate and alternate during the solution process. The most randomized operators, namely, SRP and SWP, tend to dominate at the initial stages of the search, this can be seen especially for instance *ITC2007-9* (Fig. 3). This is consistent with the known fact that search should be initially more explorative, when good solutions have not yet being located. The operators proposed in this article, namely, DDP and SDP, proved very effective in the solving process. On some instances such as *ITC2007-9* (Fig. 3) operators alternate, but a single operator dominates during a period of search process. Other instances such as *ITC2007-13* (Fig. 4), seem to require a different sequence, and a faster alternation of heuristics. Understanding these operator profiles and their correlation with specific instance features, deserves further study.

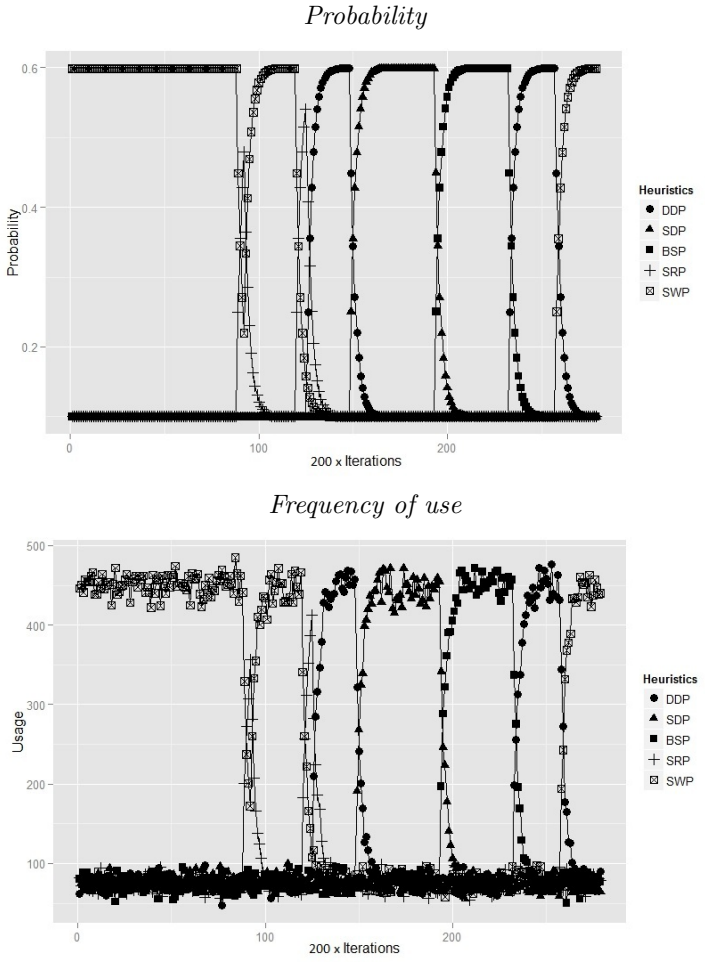

Figure 3: Instance *ITC-2007-9* operator's learned probabilities (top plot) and frequency of use (bottom plot) across a hyper-heuristic run.

5. Conclusions

The main contribution of this article is a highly-automated approach to course timetabling, obtained by combining a generic modeling approach with an adaptive search methodology incorporating learning mechanisms and effective move operators. The approach is simultaneously general and effective; general, in that different types of course timetabling policies and constraints can be modeled; effective, in that a number of move operators of different characteristics and strengths are combined into an adaptive hyper-heuristic approach, producing state-of-the art results.

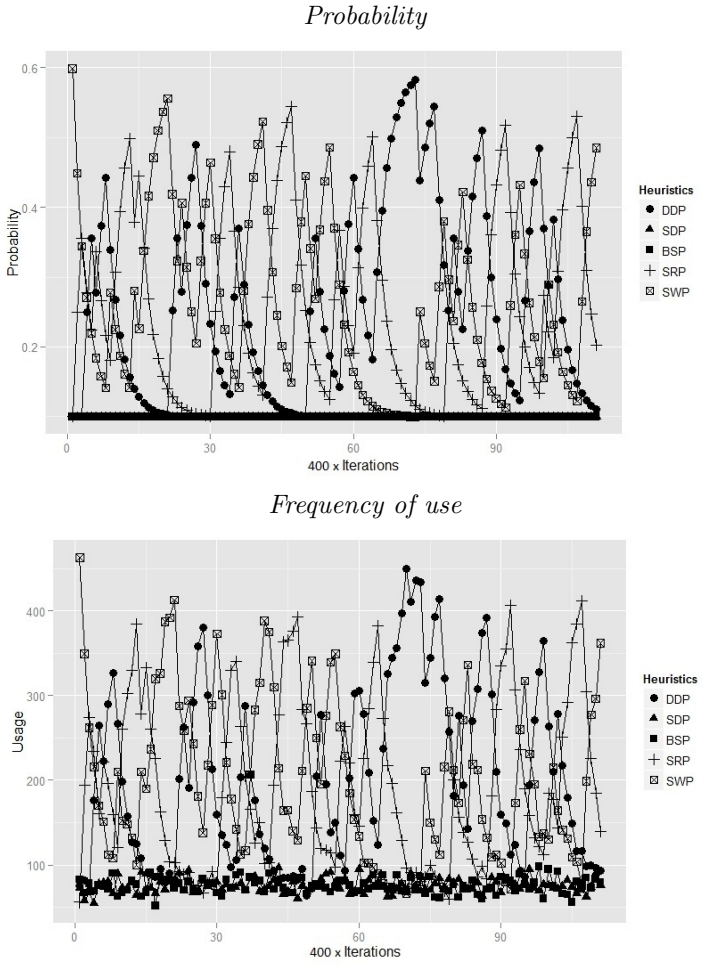

Figure 4: Instance *ITC-2007-13* operator's learned probabilities (top plot) and frequency of use (bottom plot) across a hyper-heuristic run.

The proposed hyper-heuristic approach is is based on the simple yet powerful iterated local search principle, which alternates between improving and perturbation stages. Adaptability is achieved by incorporating learning mechanisms (also called adaptive operator selection) to learn and apply alternative move operators in the improvement stage, selected from the available pool according to their past performance. The methodology also benefits from novel effective move operators, and an empirical approach for selecting the best performing operators from a larger available pool.

Several learning mechanisms, including online (dynamic) and offline (static) approaches, were implemented and tested. The best performing mechanism

resulted an online mechanism that combines extreme value credit assignment with an adaptive pursuit selection rule. This best-performing hyper-heuristic produced competitive results as compared to the the state-of-the-art on the 2007 International Timetabling Competition instances, even producing a new best-known solution.

An analysis of the learned probabilities and selection frequency of the operators in the pool reveals that different instances have different operator profiles. It is clearly demonstrated that operators cooperate and complement each other in the process of effectively solving an instance. There is no single best operator across the search process: some operators dominate at certain stages while others take over at different stages. This justifies and confirms the advantages of using online adaptive mechanisms. While there is no clear pattern across all instances, in general operators exhibiting higher degrees of randomness are preferentially selected at the early stages of the search. This is consistent with the expectation that at early stages, the search should be more explorative.

Future work will explore more sophisticated online learning mechanism, and will implement additional move operators. Population-based adaptive approaches will also be implemented and tested. Understanding and matching the effectiveness of particular move operators to particular instances and stages of the search process is an open area that requires further investigation. Finally, since the proposed adaptive high-level strategy operates in a domain-independent manner, it can be easily adapted to other timetabling, scheduling and routing problems subject to available problem-specific models and operators.

Acknowledgement

This work was supported by Consejo Nacional de Ciencia y Tecnología (CONACYT) México under PhD Scholarship 311849 and the Engineering and Physical Sciences Research Council (EPSRC - grant number EP/J017515), UK.

References

Abdullah, S., Burke, E. K., McCollum, B., 2007. Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In: Metaheuristics. Vol. 39 of Operations Research Computer Science Interfaces Series. Springer US, pp. 153–169.

Adriaen, M., De Causmaecker, P., Demeester, P., Vanden Berghe, G., 2006. Tackling the university course timetabling problem with an aggregation approach. In Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno. 1, 330–335.

Battiti, R., Brunato, M., Mascia, F., 2007. Reactive Search and Intelligent Optimization. Vol. 45 of Operations Research/Computer Science Interfaces Series. Springer.

Birattari, M., 2009. Tuning Metaheuristics: A Machine Learning Perspective. Springer.

Burke, E. K., Curtois, T., 2014. New approaches to nurse rostering benchmark instances. European Journal of Operational Research (to appear).

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society (JORS) 64 (12), 1695–1724.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J. R., 2010. Handbook of Metaheuristics. Vol. 146 of International Series in Operations Research & Management Science. Springer, Ch. A Classification of Hyper-heuristic Approaches, pp. 449–468, chapter 15.

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

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.

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

Carter, M. W., 1986. OR practice - A survey of practical applications of examination timetabling algorithms. Operations Research 34(2), 193–202.

Ceschia, S., Di Gaspero, L., Schaerf, A., 2012. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. Computers & Operations Research 39 (7), 1615 – 1624.

Chakhlevitch, K., Cowling, P., 2005. Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In: Evolutionary Computation in Combinatorial Optimization. Vol. 3448 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 23–33.

Conant-Pablos, S. E., Magaña-Lozano, D. J., Terashima-Marín, H., 2009. Pipelining memetic algorithms, constraint satisfaction, and local search for course timetabling. MICAI Mexican international conference on artificial intelligence 1, 408–419.

Derrac, J., Garcia, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1 (1), 3 – 18.

Doksum, K., 1967. Robust procedures for some linear models with one observation per cell. Annals of Mathematical statistics 38, 878–883.

Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M., 2008. Extreme value based adaptive operator selection. In: Parallel Problem Solving from Nature PPSN X. Vol. 5199 of LNCS. Springer, pp. 175–184.

García, S., Fernández, A., Luengo, J., Herrera, F., 2010. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining. Experimental analysis of power, Information sciences. 180, 2044–2064.

Gent, I. P., MacIntyre, E., Presser, P., Smith, B., Walsh, T., 1996. An empirical study of dynamic variable ordering heuristics for the CSP. In: Freuder, E. (Ed.), Principles and Practice of Constraint Programming CP96. Vol. 1118 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 179–193.

Goldberg, D. E., 1990. Probability matching, the magnitude of reinforcement, and classifier system bidding. Machine Learning 5(4), 407–425.

Hoos, H. H., 2012. Automated Algorithm Configuration and Parameter Tuning. In Autonomous Search. Springer Berlin Heidelberg, Ch. 3, pp. 37–71.

Hutter, F., Hoos, H. H., Leyton-Brown, K., Stützle, T., 2009. Paramils: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research 36 (1), 267–306.

Jat, S. N., Yang, S., 2011. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. Journal of Scheduling 14(6), 617–637.

Lewis, R., 2008. A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30 (1), 167–190.

Lewis, R., 2012. A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. Annals of Operations Research 194, 273–289.

Lourenço, H., Martin, O., Stützle, T., 2003. Iterated local search. In: Glover, F., Kochenberger, G., Hillier, F. S. (Eds.), Handbook of Metaheuristics. Vol. 57 of International Series in Operations Research & Management Science. Springer New York, pp. 320–353.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., Burke, E. K., 2010. Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing 22 (1), 120–130.

Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G., July 2010. Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In: IEEE Congress on Evolutionary Computation (CEC), 2010. pp. 1–8.

Misir, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G., 2013. An investigation on the generality level of selection hyper-heuristics under different empirical conditions. Applied Soft Computing 13 (7), 3335 – 3353.

Nannen, V., Eiben, A. E., 2006. A method for parameter calibration and relevance estimation in evolutionary algorithms. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. GECCO '06. ACM, New York, NY, USA, pp. 183–190.

Nonobe, K., Ibaraki, T., 2001. An improved tabu search method for the weighted constraint satisfaction problem. INFOR 39 (2), 131–151.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J. A., Walker, J., Gendreau, M., Kendall, G., Parkes, A. J., Petrovic, S., Burke, E. K., 2012a. Hyflex: a benchmark framework for cross-domain heuristic search. In: Proceedings of the 12th European conference on Evolutionary Computation in Combinatorial Optimization,

EvoCOP'12. Vol. 7245 of Lecture Notes in Computer Science. Springer-Verlag, pp. 136–147.

Ochoa, G., Walker, J., Hyde, M., Curtois, T., 2012b. Adaptive evolutionary algorithms and extensions to the hyflex hyper-heuristic framework. In: Parallel Problem Solving from Nature - PPSN 2012. Vol. 7492 of Lecture Notes in Computer Science. Springer, Berlin, pp. 418–427.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers and Operations Research 34, 2403– 2435.

Qu, R., Burke, E. K., McCollum, B., 2009a. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. European Journal of Operational Research 198 (2), 392 – 404.

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

Rahman, S. A., Bargiela, A., Burke, E. K., Özcan, E., McCollum, B., McMullan, P., 2014. Adaptive linear combination of heuristic orderings in constructing examination timetables. European Journal of Operational Research 232 (2), 287 – 297.

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

Sørensen, M., Dahms, F. H., 2014. A two-stage decomposition of high school timetabling applied to cases in Denmark. Computers & Operations Research 43 (0), 36 – 49.

Soria-Alcaraz Jorge, A., Carpio, M., Puga, H., Sotelo-Figueroa, M., 2013a. Comparison of Metaheuristic Algorithms with a Methodology of Design for the Evaluation of Hard Constraints over the Course Timetabling Problem. Vol. 451 of Studies in Computational Intelligence. Springer Berlin Heidelberg.

Soria-Alcaraz Jorge, A., Carpio, M., Puga, H., Terashima-Marin, H., Cruz Reyes, L., Melin-Olmeda, E., Sotelo-Figueroa, M. A., 2013b. Methodology of Design: A novel generic approach applied to the Course Timetabling problem. Vol. 451 of Studies in Computational Intelligence. Springer Berlin Heidelberg.

Soria-Alcaraz Jorge, A., Terashima-Marin, H., Carpio, M., 2010. Academic timetabling design using hyper-heuristics. Advances in Soft Computing, ITT Springer-Verlag 1, 158–164.

Thierens, D., 2005. An adaptive pursuit strategy for allocating operator probabilities. In: Proceedings of the 2005 conference on Genetic and evolutionary computation. GECCO '05. ACM, New York, NY, USA, pp. 1539–1546.

Walker, J., Ochoa, G., Gendreau, M., Burke, E. K., 2012. Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: Hamadi, Y., Schoenauer, M. (Eds.), Learning and Intelligent Optimization. Vol. 7219 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 265–276.