Soft Computing manuscript No.
(will be inserted by the editor)

A Greedy Gradient-Simulated Annealing Selection Hyper-heuristic

Murat Kalender · Ahmed Kheiri · Ender Özcan · Edmund K. Burke

Received: date / Accepted: date

Abstract Educational timetabling problem is a challenging real world problem which has been of interest to many researchers and practitioners. There are many variants of this problem which mainly require scheduling of events and resources under various constraints. In this study, a curriculum based course timetabling problem at Yeditepe University is described and an iterative selection hyper-heuristic is presented as a solution method. A selection hyper-heuristic as a high level methodology operates on the space formed by a fixed set of low level heuristics which operate directly on the space of solutions. The move acceptance and heuristic selection methods are the main components of a selection hyper-heuristic. The proposed hyper-heuristic in this study combines a simulated annealing move acceptance method with a learning heuristic selection method and manages a set of low level constraint oriented heuristics. A key goal in hyper-heuristic research is to build low cost methods which are general and can be reused on unseen problem instances as well as other problem domains desirably with no additional human expert intervention. Hence, the proposed method is additionally applied to a high school timetabling problem, as well as six other problem domains from a hyper-heuristic benchmark to test its level of generality. The empirical results show that our easy-to-implement hyper-heuristic is effective in solving the Yeditepe course timetabling problem. Moreover, being sufficiently general, it delivers a reasonable performance across different problem domains.

Keywords Heuristic · Hyper-heuristic · Timetabling · Computational Design

The initial version of this study was presented at UKCI 2012: 12th Annual Workshop on Computational Intelligence

Murat Kalender
Yeditepe University, Computer Engineering Department
Inonu Mh., Kayisdagi Cd., 34755 Kadikoy/Istanbul, Turkey
E-mail: mkalendertr@yahoo.com

Ahmed Kheiri, Ender Özcan
University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
Tel.: +44-115-9514234, 9515544
Fax: +44-115-9514254
E-mail: {axk, exo}@cs.nott.ac.uk

Edmund K. Burke
University of Stirling
Cottrell Building, Stirling FK9 4LA, UK
Tel.: +44-1786-467020
Fax: +44-1786-467016
E-mail: e.k.burke@stir.ac.uk

1 Introduction

A *hyper-heuristic* is a high level search methodology which performs a search over the space of heuristics rather than the space of solutions for solving hard computational problems (Burke et al, 2013). The idea of combining different heuristics (neighbourhood operators) with the goal of exploiting their strengths dates back to the early 1960s (Fisher and Thompson, 1963; Crowston et al, 1963). Since then, there has been a growing interest into hyper-heuristics. A recent theoretical study shows that mixing heuristics could lead to exponentially faster search than using each standalone heuristic on some benchmark functions (Lehre and Özcan, 2013). Hyper-heuristics that control and mix a fixed set of low level heuristics are referred to as *selection hyper-heuristics*. A selection heuristic generally combines a *heuristic selection* and *move acceptance* methods under an iterative framework. At each step, a low level heuristic is used to modify a solution in hand, then a decision is made whether to accept or

reject the new solution. Almost all previously proposed selection hyper-heuristics are designed respecting the concept of a *domain barrier* which separates the hyper-heuristic from the problem domain containing the low level heuristics (Cowling et al, 2001) as illustrated in Figure 1. Traditionally, this barrier prohibits any problem domain specific information to pass through to the hyper-heuristic level. This type of layered and modular approach to the design of automated search methodologies supports the development of more general methods than currently there exist, which are applicable to unseen instances from a single problem domain or even different problem domains. Moreover, reuse of algorithmic components becomes possible.

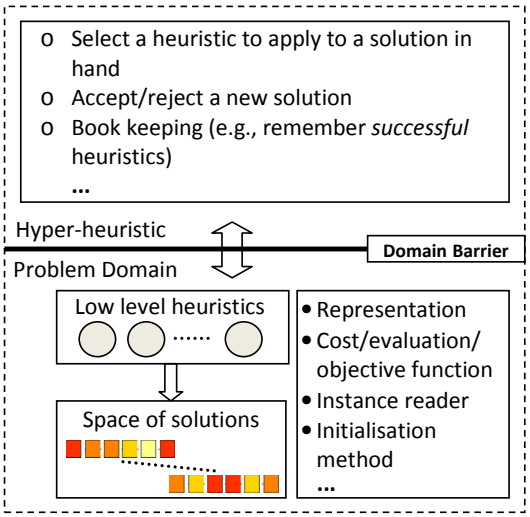

Fig. 1 A selection hyper-heuristic framework.

Educational timetabling problem is a challenging real-world combinatorial optimisation problem which is known to be NP-hard (Even et al, 1976; de Werra, 1997). There are different types of educational timetabling problems, such as university course timetabling and high school timetabling which have been of interest to many researchers and practitioners. This study mainly concerns university course timetabling (Lewis, 2007; Lewis et al, 2007; Erben and Keppler, 1996; Socha et al, 2002). Two subclasses of university course timetabling problems can be identified in the literature: (i) *post-enrolment problems* in which the student enrolment is known, (ii) *curriculum based problems* in which the student enrolment is not known, but curriculums of the students are available prior to the timetabling process (McCollum et al, 2010). A solution to a given university course timetabling problem requires scheduling of courses considering limited resources subject to a set of *hard* and *soft* constraints. In most of the cases, a *feasible* solution which satisfies the hard constraints is sought. The soft constraints represent preferences. A solution method attempts to satisfy as many of the soft constraints as possible.

In this study, a curriculum based university course timetabling problem constantly dealt with at Yeditepe University, Faculty of Engineering and Architecture, Computer Engineering Department is introduced. Additionally, a selection hyper-heuristic solution to the problem, which combines a fast reacting greedy and gradient heuristic selection mechanism with a simulated annealing is described. We compared the performances of different heuristic selection methods used within a selection hyper-heuristic framework. In order to show that the proposed hyper-heuristic is sufficiently general and can be applied to the other problem domains without requiring any change, it is implemented as an extension to a public software library and tested on a high school timetabling problem as well as a hyper-heuristic benchmark. The empirical results indeed show that our hyper-heuristic is adaptive and general, performing better than some previously proposed approaches on university course timetabling problem, high school timetabling problem and six other domains from the benchmark. Designing an effective and general selection hyper-heuristic approach or its component with less number of parameters to tune or control has always been of interest. The proposed heuristic selection method has no parameter to set.

An earlier version of this research first appeared in the UKCI conference (Kalender et al, 2012). Following the conference, the editors issued an invitation to submit extended versions of the conference papers to this special issue. This paper is the result of that process. The performance analysis of our approach on course timetabling and hyper-heuristic benchmark domains is revised and we report additional results obtained from testing our hyper-heuristic on a new domain, namely high school timetabling.

Section 2 provides a brief overview of the selection hyper-heuristics that relates to the design of our solution method and previous approaches used to solve university course timetabling problem. Section 3 describes the curriculum-based course timetabling problem at Yeditepe University and the developed selection hyper-heuristic framework including all algorithmic components and low level heuristics for solving it. Section 4 summarises initial set of experimental results and compares the performance of different hyper-heuristics including the proposed one on Yeditepe course timetabling. Section 5 covers the rest of the experimental results discussing the performance of the proposed hyper-heuristic on high school timetabling and

other benchmark problem domains. Finally, Section 6 presents the conclusions.

2 Related Work

2.1 Selection Hyper-heuristics

There are two main types of hyper-heuristic methodologies in the literature: methodologies to *select* and *generate* heuristics (Burke et al, 2010). This study focuses on an iterative selection hyper-heuristic framework based on a single point search consisting of two stages: *heuristic selection* and *move acceptance* (Özcan et al, 2008; Cowling et al, 2001). Firstly, a hyper-heuristic under such a framework attempts to improve a solution in hand by selecting and applying an appropriate heuristic from a fixed set of low level heuristics. This stage yields a new solution. Then, a decision is made whether to accept or reject this new solution. The search process continues iteratively until the termination criteria are satisfied. A selection hyper-heuristic controls and mixes a set of *perturbative* low level heuristics, each processing and returning a complete solution when invoked In this part, we discuss some selection hyper-heuristics and their components from the literature that relates to the design of our hyper-heuristic. A selection hyper-heuristic will be denoted as *heuristic selection−move acceptance* from this point onward.

Cowling et al (2001) investigated the performance of many *simple* selection hyper-heuristic components on a scheduling problem. The heuristic selection methods covered in this study include *simple random*, *random descent*, *random permutation*, *random permutation descent*, *greedy* and *choice function*. Greedy applies all heuristics to the current candidate solution and chooses the one that achieves the best quality. Choice function utilises a mechanism that scores each heuristic based on its individual performance, pair-wise successive performance and the duration since the last time a heuristic was invoked. At each step, choice function selects a heuristic with the maximum score and updates the relevant information for the chosen heuristic after its application to the current solution. A hyper-heuristic either utilises a learning mechanism or operates without any learning at all (Burke et al, 2010). Both greedy and choice function are online learning methods, since they get feedback during the search process. The memory length of choice function is determined by means of the limits on the score values. A larger range for the score indicates a longer term memory as compared to a lower range. On the other hand, greedy has the shortest memory and gets instantaneous feedback during the search process, then forgets this feedback in the following step. Cowling et al (2001) combined these heuristic selection methods with two move acceptance strategies including *accept all moves* and *accept only improving moves*. Cowling et al (2001) reported that choice function−accept all moves is the most promising hyper-heuristic. The successful performance of the choice function heuristic selection method has also been confirmed by the other studies (Özcan et al, 2008; Burke et al, 2012; Bilgin et al, 2007).

There are different types of move acceptance methods used within selection hyper-heuristics in the literature (Burke et al, 2013). Mostly, those methods accept all improving moves, but they differ at how they treat non-improving moves. For example, simulated annealing move acceptance method accepts non-improving moves with a probability provided in Equation 1.

$$p_t = e^{-\frac{\Delta f}{\Delta F(1-\frac{t}{T})}} \quad (1)$$

where Δf is the quality change at step t, T is the maximum number of steps, ΔF is an expected range for the maximum quality change in a solution after applying a heuristic. (Bai and Kendall, 2005; Bai et al, 2007b; Bilgin et al, 2007) reported the success of simulated annealing as a move acceptance on the shelf allocation and examination timetabling problems, respectively. Moreover, Bilgin et al (2007) tested 36 different hyper-heuristics by pairing up a range of heuristic selection and move acceptance methods over a set of examination timetabling problem instances. The results indicate the success of the choice function−simulated annealing hyper-heuristic.

2.1.1 Hyper-heuristics Flexible Framework

Hyperion (Swan et al, 2011) and Hyper-heuristics Flexible Framework (Hyflex) (Ochoa et al, 2012) are recent software libraries which are made publicly available for rapid development of hyper-heuristics (as well as metaheuristics) and research. The Java Hyflex implementation provides an object-oriented hyper-heuristic framework, having support for six minimisation problem domains of Boolean Satisfiability (SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Hyflex strictly imposes the domain barrier and does not give user any access to the problem domain dependant information (see Figure 1). Hyflex was recently used at the Cross-Domain Heuristic Search Challenge (CHeSC 2011) [1]. The goal of this competition was determining the best selection hyper-heuristic with the

[1] http://www.asap.cs.nott.ac.uk/chesc2011/

best mean performance across thirty problem instances, five from each of the six problem domains. 20 competitors reached the finals in the competition. CHeSC 2011, including the Hyflex implementation and competing hyper-heuristics, currently serves as a benchmark to compare the performance of selection hyper-heuristics.

Hyflex provides implementation of each domain with a set of low level heuristics. Hyflex low level heuristics are classified as mutational (MU), hill climbing (HC), ruin and re-create (RC) and crossover (XO) heuristics. All heuristics are perturbative. A mutational heuristic returns as solution after processing a given solution with no quality guarantee, while a hill climbing heuristic always returns a non-worsening solution, even if the returned solution is the same as the input. Ruin and re-create heuristic first creates a partial solution based on a given solution and then rebuild a complete solution. The crossover low level heuristics take two solutions as a parameter, combine them and return a new solution. The number of the low level heuristics for each heuristic/operator type for each problem domain implemented in Hyflex is summarised in Table 1. We use OPid to denote the id^{th} low level heuristic of type OP. For example, MU0 and MU5 for SAT are the 0^{th} and 5^{th} mutational low level heuristics in the SAT domain.

Table 1 The number of different types of low level heuristics {mutation (MU), hill climbing (HC), ruin and re-create (RC), crossover (XO)} used in each problem domain.

Domain	MU	HC	RC	XO	Total
SAT	6	2	1	2	11
BP	3	2	2	1	8
PS	1	5	3	3	12
PFS	5	4	2	4	15
TSP	5	3	1	4	13
VRP	3	3	2	2	10

2.2 University Course and High School Timetabling Problems

Due to the intrinsic difficulty of educational timetabling problems (Even et al, 1976; de Werra, 1997), the exact solvers generally fail to produce high quality solutions in a given time. Hence, alternative approaches have been used to solve university course and high school timetabling problems, ranging from single point based search methods, including simulated annealing and tabu search to population based methods, such as, evolutionary algorithms and ant colony optimisation. High school timetabling is different from university course timetabling. The main difference is that the timetable for a student is more packed in high schools and students are fully occupied throughout a day. Consequently, the shared resources are more loaded.

Abramson (1991) employed simulated annealing for course timetabling. Colorni et al (1992) investigated the performances of genetic algorithm, simulated annealing and tabu search. They observed that memetic algorithm combining genetic algorithm and local search performed better. Hertz (1992) utilised tabu search. Erben and Keppler (1996) employed genetic algorithms with smart operators to generate a weekly timetable with a heavily constraint problem instance. Binary encoding is used as a representation scheme. Schaerf (1996) used tabu search to solve high-school course timetabling problems and developed an interactive interface. Paechter et al (1998) used an evolutionary algorithm and developed a user interactive tool which allowed users to visualise violated objectives and modify the objectives during a run for solving Napier University timetabling problem. Abramson et al (1999) tested different cooling schedules within simulated annealing for course timetabling. Filho et al (2001) formulated timetabling problem as a clustering problem and applied a constructive genetic algorithm for solving timetabling problems of public schools in Brazil. Socha et al (2002) described a max-min ant system for solving course timetabling problem and compared their approach to a random restart local search approach using eleven benchmark problem instances.

Alkan and Özcan (2003) hybridised a violation directed hierarchical hill climbing method (VDHC) using constraint oriented neighbourhood heuristics with genetic algorithms for solving the university course timetabling problem. Similarly, the constraint oriented neighbourhood heuristics were found to be effective when used as a part of a hybrid framework in Özcan et al (2012) for solving a variant of a high school course timetabling problem. Burke et al (2003) used a combination of tabu search and reinforcement learning scheme as a heuristic selector and tested their hyper-heuristic over different timetabling problems. Burke et al (2006) employed a case-based reasoning approach as a hyper-heuristic using different measures for similarity of instances for solving course timetabling problems. Burke et al (2003) used tabu search hyper-heuristic to build solutions using graph colouring heuristics for solving timetabling problems.

2.2.1 International Timetabling Competition

Determining the state-of-the-art method among modern approaches for a given timetabling problem and providing a real world benchmark for comparison of

approaches are the main deriving ideas behind the International Timetabling Competition series. ITC2007 (McCollum et al, 2010) hosted by PATAT and WATT[2] was on educational timetabling. Recently, the Third International Timetabling Competition (ITC2011)[3] on high school timetabling with three different rounds was organised. In this study, we test our hyper-heuristic on the high school timetabling problem instances obtained used in the second round of this competition. Moreover, we compare its performance to the competing algorithms. The ITC2011 problem instances consist of (i) *times* which are indivisible time intervals (ii) *resources* that attend the events (iii) *events* which indicate the meetings between resources and 15 types of (iv) *constraints*, including *assign resource, assign time, split events, distribute split events, prefer resources, prefer times, avoid split assignments, spread events, link events, avoid clashes, avoid unavailable times, limit idle times, cluster busy times, limit busy times, limit workload* (Post et al, 2012). Each constraint could be defined as hard or soft for a given instance. In the ITC2011 competition, hard constraints violations are relaxed and they are simply much more heavily penalised than the 'soft' constraints based on weights.

3 A Selection Hyper-heuristic Framework for Solving a Course Timetabling Problem

3.1 Problem Description

Every year, Computer Engineering Department (and so the other departments as well) at Yeditepe University, Faculty of Engineering and Architecture deals with a curriculum-based course timetabling problem. Each student has to follow a curriculum at Yeditepe University. Since, time to time some changes are made to the curriculums, there might be a cohort of students with different curriculums to follow based on the existing courses at a given time. A curriculum consists of eight terms and there are on average six courses per term for a student to register. In general, a student registers to all the courses at a given term, unless the student has failed from some previous courses. The latter type of students are not considered during the timetabling process. Some courses have prerequisites and/or co-requisites. The timetables are produced for the regular students. A course consists of lectures, problem solving and/or laboratory session meetings which could take place at different locations (rooms). It is always desirable that the lab or problem solving sessions are after the lecture hours for a given course.

Lecturers handle the teaching, while laboratory and problem solving sessions could be handled by a lecturer or a teaching assistant or both. There are full time and part time lecturers. The requests of part time lecturers regarding the time that they teach have to be accommodated. There are some courses which have to be taken by all students across the university and by all engineering students and by all students at a department. Similarly, there are optional courses open to all students in the university, or within the faculty, or within a department. Moreover, the optional courses are part of the curriculum appearing in different terms. A lecture, problem solving or laboratory session meeting takes 1, 2 or 3 hours, respectively. The university imposes a template for the other units to follow to make the timetabling process easier as illustrated Figure 2. Only certain slots can be allocated for the meetings of 1 and 2 hour duration. 3 hour meetings have to consist of (2+1, 1+2) blocks. The lecturers are allowed to provide preference for their lectures, which is taken seriously. In general, the teaching assistants are themselves postgraduate students taking other courses, hence their teaching/tutorial hours must not overlap with the lectures that they will attend.

After the university sets the times for the university-wide compulsory courses, the faculty does the same and passes the information to the departments. Then the departments have to deal with the timetabling of the remaining courses and the required resources. The following hard constraints are identified:

- **C01:** The timetable template provided by the university must be respected while scheduling meetings (see Figure 2). 2-hour blocks cannot be divided into a 1-hour block.
- **C02:** Course meetings can be assigned to predefined time-slots.
- **C03:** A set of courses can appear as a part of multiple terms in the curriculum. This is to accommodate optional courses.
- **C04:** Course meetings can be enforced to take place in the same day or in different days.
- **C05:** (w_1) Meetings of a lecturer must not overlap.
- **C06:** (w_2) Lecturers can provide their weekly availability (or unavailability) for teaching.
- **C07:** (w_3) The courses in a given term of the curriculum must not overlap.
- **C08:** (w_4) Certain time-slots from the weekly timetable can be excluded during the timetabling process of the courses for a term. One of the uses of this constraint is to arrange a common time slot for departmental or faculty meetings. The aim is to get as

[2] http://www.cs.qub.ac.uk/itc2007/
[3] ITC2011 website: http://www.utwente.nl/ctit/hstt/

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
09:00-10:50					
11:00-11:50					
12:00-12:50					
13:00-13:50					
14:00-15:50					
16:00-16:50					
17:00-18:50					

Fig. 2 Yeditepe University course timetabling template.

many lecturers free of teaching during those times as possible, and so it is a soft constraint.
- **C09:** (w_5) A room with a suitable capacity must be allocated for each course without any overlap.
- **C10:** (w_6) The equipment required by a course must be allocated without any overlap.

Soft constraints are as follows:

- **C11:** (w_7) The duration between the meetings of a lecturer on a day should be within predefined minimum and maximum limits.
- **C12:** (w_8) The duration between the meetings on a day for a regular student (studying term) should be within predefined minimum and maximum limits.
- **C13:** (w_9) The total number of meeting hours during when a lecturer teaches on a day cannot exceed a predetermined maximum value.
- **C14:** (w_{10}) The total number of meeting hours that a regular student (studying a term) attends on a day should be within predefined minimum and maximum limits.
- **C15:** (w_{11}) The total number of courses scheduled for a lecturer on a day cannot exceed a predetermined maximum value.
- **C16:** (w_{12}) The order of between different course meetings can be defined.

The constraints C01-C04 are handled through representation and restricting the value assignment to each course and so does not require any further attention during the search process. Solving the course timetabling problem requires finding a high quality timetable with the minimum number of constraint violations, if possible with no violations. The objective function (evaluation/cost) used in this study takes the weighted average of the total number of constraint violations and treats all constraints as if they were the same, but punishes the hard constraint violations heavier than the soft constraint violations:

$$objectiveFunction(T) = \sum_{\forall i} w_i g_i(T) \quad (2)$$

where T represents a candidate timetable, w_i indicates the weight associated to constraint i, g_i indicates the number of constraint violations of constraint i for the given timetable. The goal is to find a timetable which minimises the cost computed using the objective function. The cost reflects the quality of a given timetable. Lower the cost, better the quality of a timetable gets. The minimum possible cost occur whenever a *perfect solution* is obtained with an objective value of 0, indicating that there are no constraint violations. The weight values used during the experiments are provided in Table 2.

Table 2 Conflict weight values of the hard and soft constraints

w_1	w_2	w_3	w_4	w_5	w_6
5	5	5	5	3	3
w_7	w_8	w_9	w_{10}	w_{11}	w_{12}
1	1	3	3	1	1.9

3.2 A Selection Hyper-heuristic Framework Using Greedy Gradient Heuristic Selection

In this study, we present a learning hyper-heuristic for solving curriculum-based course timetabling problem at Yeditepe University. In most of the previous applications of reinforcement learning in hyper-heuristics, a utility value is increased as a reward mechanism and decreased for punishment (Nareyek, 2004; Bai et al, 2007a). It has also been observed that the memory length affects the performance. The proposed hyper-heuristic framework is somewhat adapts a similar strategy. Instead of a predefined scoring mechanism, the cost change in between the old and current solution generated after the application of the selected heuristic is used as a utility value. Whenever the utility value of each heuristic is 0, a greedy-like strategy is invoked (Algorithm 1, steps 2, 3 and 4). Each heuristic is called one by one using the same solution at hand and the cost change is recorded as a utility value of the corresponding heuristic. If a heuristic causes a worsening move, its utility value is set to 0. Then, a heuristic is chosen based on the scores (Algorithm 1, steps 6 and 7). In this study, *max* function which chooses an option with the highest value and in this case, chooses a heuristic with the maximum score is employed. After applying the selected heuristic, its score is updated right away using the cost change. This strategy neither makes use of a periodic update of scores as in (Bai et al, 2007b), nor forgets the scores as soon as a heuristic is selected as in a greedy method (Cowling et al, 2001). In the case when one heuristic has a non-zero value, it will be selected as

long as the solution improves and the hyper-heuristic will act like a gradient hill climber.

During the heuristic selection process, utility values of a subset of heuristics returned by the *max* function might be the same, necessitating a tie breaking strategy. Two different cases emerge: a non-zero tie score for some heuristics or all 0s. A random selection is performed in the former case. For the latter case, a problem dependent feature is implemented. Another utility array is maintained to keep track of the number of violations due to each constraint type. Again, *max* function is used for determining the highest number of violations and the corresponding constraint type. Hence, the corresponding heuristic is invoked. Then, the utility values of the selected heuristic are updated in both arrays using the new solution.

Algorithm 1 Pseudocode of the greedy gradient heuristic selection method

```
1: procedure GG_Select_Heuristic(scores, current solution)
2:     if all heuristic scores are 0 then
3:         invoke each heuristic using the current solution
4:         record cost change as the score for each heuristic
5:         reset the score of a heuristic to 0 if cost increases
6:     end if
7:     choose a heuristic based on the scores
8:     in case of a tie, use a tie breaking strategy
9:     return (chosen heuristic id for invocation)
10: end procedure
```

A hyper-heuristic framework is implemented using the simulated annealing move acceptance method which allows non-improving moves based on equation 1 (see section 2.1). In some problem domains, the maximum (expected) change in the quality of solutions are not easy to be estimated. In such cases, ΔF is set to a factor of the cost of the best solution in hand. There is strong empirical evidence showing that the choice of a selection hyper-heuristic components influences its performance (Özcan et al, 2008; Özcan et al, 2006). In this study, the performances of different heuristic selection methods including the greedy gradient under the simulated annealing based selection hyper-heuristic framework are investigated. This framework contains a fixed set of constraint based neighbourhood operators as low level heuristics, similar to the ones designed in Alkan and Özcan (2003), and Özcan and Ersoy (2005). Each low level heuristic attempts to improve upon a corresponding constraint. An event (course) causing the relevant violation is rescheduled to the best timeslot which reduces the overall cost at most. For example, if the low level heuristic handling C15 violation is selected, then one of the events (courses) causing that violation is randomly chosen and rescheduled to the timeslot which produces the least cost. Selection hyper-heuristics work as a high level strategy to manage those low level heuristics. They aim to find a solution attempting to minimise the hard and soft constraint violations, simultaneously.

4 Experimental Results for the Yeditepe Course Timetabling Problem

The performance of four selection hyper-heuristics are investigated over eight instances (rp1-8) which are randomly generated based on the Yeditepe course timetabling problem and a real instance (*cse*). The experiments were performed on a PC P4 Processor 3 GHz, 512 RAM. A run terminates after solution found or time limit reached 600 seconds. Figure 3 summarises the performance of each hyper-heuristic based on 50 runs for each instance. As evaluation measure *success rate* is used: $s.r.$ = (number of runs for which the perfect solution is obtained)/50. The rankings of the different hyper-heuristics in Figure 3 are calculated according to the success rates, the average best cost and the average best duration values of the tests. Lower the ranking, better a hyper-heuristic is.

The results show that greedy gradient, in the overall, performs better than simple random (SR), greedy (GR) and choice function (CF) heuristic selection methods as a part of a selection hyper-heuristic embedding simulated annealing (SA) as a move acceptance method. It is successful in particular when the problem size grows. For the *cse* instance, all hyper-heuristics perform similarly. Our ultimate goal was to be able to solve the university timetabling problem for the whole university when the solver was designed. The results show that greedy gradient−simulated annealing (GG−SA) hyper-heuristic is promising in this respect.

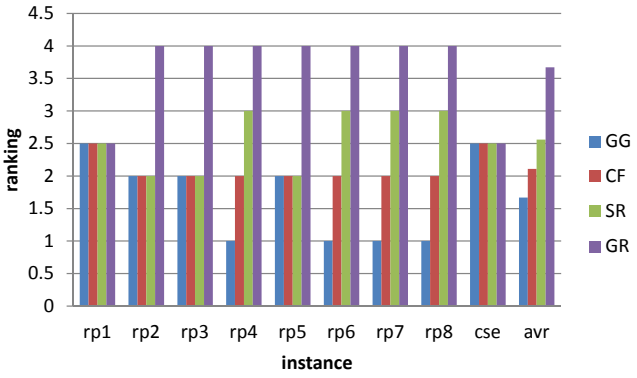

Fig. 3 Performance ranking of each hyper-heuristic combined with the SA move acceptance over the Yeditepe course timetabling instances

Table 3 provides a pairwise performance comparison of two top ranking selection hyper-heuristics, the greedy gradient−simulated annealing (GG−SA) and the choice function−simulated annealing (CF−SA) based on average cost using the Wilcoxon singed-rank test on the Yeditepe course timetabling instances. The following notation is used: Given A versus B, $>$ ($\geq$) denotes that A performs (slightly) better than B, since this performance variation is (not) statistically significant within a 95% confidence level, while $\simeq$ indicates that they deliver the same performance. Greedy gradient−simulated annealing performs significantly better than choice function −simulated annealing on three instances: rp4, rp6, rp7. Greedy gradient−simulated annealing is slightly better than choice function−simulated annealing for rp1 and cse, while they perform the same for the rest of the instances.

Table 3 Average performance comparison between GG−SA and CF−SA.

label	GG−SA vs. CF−SA
rp1	GG−SA $\geq$ CF−SA
rp2	GG−SA $\simeq$ CF−SA
rp3	GG−SA $\simeq$ CF−SA
rp4	GG−SA $>$ CF−SA
rp5	GG−SA $\simeq$ CF−SA
rp6	GG−SA $>$ CF−SA
rp7	GG−SA $>$ CF−SA
rp8	GG−SA $\simeq$ CF−SA
cse	GG−SA $\geq$ CF−SA

In most of the cases, the hyper-heuristics rapidly improve the quality of the solutions in hand. After a while, the improvement process slows down as the approach reaches a local optimum. Still, it seems that the simulated annealing acceptance works well as a part of the implemented hyper-heuristics, allowing further improvement in time until we get the perfect solution where all the constraints are satisfied. This behaviour is illustrated in Figure 4. The figure shows the average best cost over 50 runs versus the time in seconds for greedy gradient−simulated annealing and choice function− simulated annealing on the rp8 instance.

5 Performance of Greedy Gradient−Simulated Annealing on the Other Problem Domains

The experimental results on the Yeditepe course timetabling problem indicates the success of greedy gradient −simulated annealing. This section provides the results of the experiments in which we have tested the performance of this hyper-heuristic on high school timetabling as well as six other problem domains from the CHeSC 2011 benchmark.

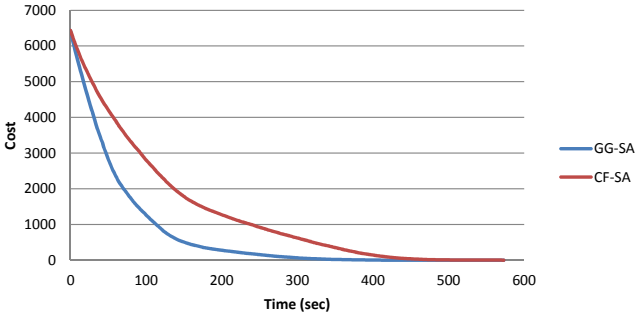

Fig. 4 Average best cost (over 50 runs) versus time for the top two hyper-heuristics on rp8

5.1 High School Timetabling

Greedy gradient−simulated annealing is tested on high school timetabling problem instances from the ITC2011 dataset which contains a variety of instances obtained from different countries across the world. A solution is evaluated using concatenation of *infeasibility* and *objective* values ($infeasibility - value.objective - value$) as cost which represent the weighted hard and soft constraint violations, respectively. For example, a cost value of 63.00225 indicates an infeasibility value of 63 and objective value of 225. In the second round of the competition, each algorithm was given 1000 nominal seconds with respect to the organisers' computer. Ten trials were performed using each algorithm for each instance, and then algorithms were ranked for each result. The average ranking was used to determine the winner. Four solvers, each identified by the name of the designing team were submitted to the ITC2011 competition (Post et al, 2012). HySST (Kheiri et al, 2012) applied a multi-stage hyper-heuristic managing a set of mutational heuristics and two hill climbers. This selection hyper-heuristic incorporates random choice for the heuristic selection and an adaptive threshold move acceptance method. HFT (Domrös and Homberger, 2012) used an evolutionary algorithm as a solution method. Lectio (Sørensen et al, 2012) employed an approach based on adaptive large neighbourhood search. GOAL (Fonseca et al, 2012) combined iterated local search based on multiple neighbourhood operators with simulated annealing, which turned out to be the winner of the second round of the competition.

In this study, greedy gradient−simulated annealing manages the same set of seven mutational low level heuristics as used by HySST (Kheiri et al, 2012). A low level heuristic swaps, combines, splits or reschedules events, times or resources, randomly. The ΔF value in

the simulated annealing move acceptance component of our hyper-heuristic is set to 0.1 of the cost of the best solution in hand if there is any hard constraint violation. This value is set to 0.0001, if the best solution contains only soft constraint violations, for which the infeasibility value is 0.

Based on the same performance measurement and rules of the competition, greedy gradient−simulated annealing ranks the second among the competing algorithms as illustrated in Table 5.1. GOAL is still the best ranking approach generating the best solutions for twelve instances. Greedy gradient−simulated annealing achieves the best results for the ElementarySchool - Finland and Kottenpark2003 - Netherlands instances. Moreover, the Wilcoxon singed-rank test reveals that greedy gradient−simulated annealing performs significantly better than GOAL and HySST on average within a confidence interval of 95% on four instances of Instance2 - Brazil, Instance6 - Brazil, ElementarySchool - Finland and Woodlands2009 - South Africa. This superior performance over HySST is observed on Instance1 - Kosovo and on Instance4 - Brazil over GOAL. The average performance of the GG−SA hyper-heuristic is slightly better on Kottenpark2003 - Netherlands than GOAL, and HySST on SecondarySchool2 - Finland, Instance4 - Brazil and Kottenpark2003 - Netherlands.

5.2 CHeSC 2011 Benchmark

Greedy gradient−simulated annealing is further tested on Hyflex problem domains, used in an earlier competition leading to the main event and CHeSC 2011. In both competitions, a hyper-heuristic was given ten nominal minutes to run. All algorithms were then ranked using the Formula1 scoring system. In this system, the best performing hyper-heuristic for a given instance receives an award of 10 points, while the second one gets 8, and the next one gets 6, 5, 4, 3, 2, 1 in that order and then all the remaining approaches get zero point. These points are accumulated as a score for a hyper-heuristic over all instances from all problem domains. The proposed hyper-heuristic is implemented as an extension to Hyflex. In the simulated annealing acceptance method, the value of ΔF is fixed as 0.01 of the cost of the best solution in hand, since Hyflex does not have a feature which supports the computation of maximum (expected) change in the quality of solutions. Our hyper-heuristic operates under a single point based search framework, and so ignores all crossover operators during the search process for any given problem. We use Formula1 scoring system for comparing the performance of our hyper-heuristic against the other algorithms in both competitions.

5.2.1 Comparison to the mock competition hyper-heuristics

Prior to the actual competition of CHeSC 2011, the organisers arranged a mock competition using eight hyper-heuristics (HH1-HH8). The results of this mock competition were provided for the competitors to form a baseline and assess the performance of their algorithms. The mock competition hyper-heuristics were designed based on the previously proposed techniques from the literature. All hyper-heuristics were allowed to run a single trial on 10 instances from each problem domain, including boolean satisfiability (SAT), one dimensional bin packing (BP), permutation flow shop (PFS) and personnel scheduling (PS), given ten nominal minutes as the time limit. Then eight hyper-heuristics were ranked based on the Formula1 scoring system. The maximum overall score that a hyper-heuristic can achieve was 400. More details on the mock competition can be found at the CHeSC 2011 website.

In the SAT problem domain, our hyper-heuristic produces the best results in 3 out of 10 instances and there is a tie in 1 instance when compared to the mock competition hyper-heuristics. It is the second best hyper-heuristic based on the Formula1 scoring system in this domain. In the bin packing problem domain, our hyper-heuristic performs still well and produces the best results in 2 instances, but in the personnel scheduling problem, its performance is not as good as on the other problem domains. In permutation flow shop, the proposed hyper-heuristic produces the best results in 2 instances. Figure 5 provides the individual and overall ranking of our hyper-heuristic for each problem domain based on Formula1 scoring system and overall the proposed hyper-heuristic ranks the third with a total score of 211.5.

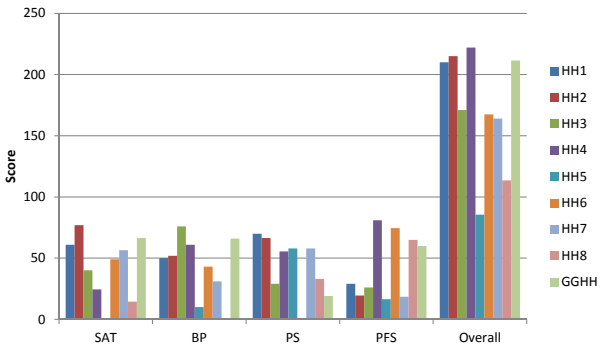

Fig. 5 Comparisons of the different hyper-heuristics over each domain based on Formula1 scores

Table 4 The characteristics of the ITC2011 dataset (where *t.*: times, *T.*: number of teachers, *R.*: number of rooms, *C.*: number of classes, *S.*: number of students, *dur.*: duration) and performance comparison of GG−SA to the other competing approaches over 10 trials showing the best quality (cost) of a solution indicated as feasibility-value.objective-value in Round 2 of ITC2011. The best values are highlighted in bold.

Problem: Country	*t.*	*T.*	*R.*	*C.*	*S.*	*dur.*	GG-SA	HySST	GOAL	HFT	Lectio
Instance2 - Brazil	25	14		6		150	0.00046	1.00069	1.00051	5.00183	**0.00019**
Instance3 - Brazil	25	16		8		200	0.00122	0.00096	**0.00087**	26.00264	0.00112
Instance4 - Brazil	25	23		12		300	1.00234	2.00238	16.00104	63.00225	**1.00172**
Instance6 - Brazil	25	30		14		350	0.00201	2.00229	4.00207	21.00423	**0.00183**
ElementarySchool - Finland	35	22	21	60		445	**0.00003**	0.00004	**0.00003**	29.00080	**0.00003**
SecondarySchool2 - Finland	40	22	21	36		566	0.00035	0.00006	**0.00000**	28.01844	0.00014
Aigio 1st HS 2010	35	37		208		532	0.00514	0.00322	**0.00006**	45.03665	0.00653
Instance4 - Italy	36	61		38		1101	0.00882	0.04012	**0.00169**	250.05966	0.00225
Instance1 - Kosovo	62	101		63		1912	71.35367	1065.17431	**38.09789**	986.42437	274.04939
Kottenpark2003 - Netherlands	38	75	41	18	453	1203	**0.18738**	0.47560	0.87084	203.87920	34.55960
Kottenpark2005A - Netherlands	37	78	42	26	498	1272	30.27471	**26.35251**	27.37026	393.40463	185.83973
Kottenpark2008 - Netherlands	40	81	11	34		1118	51.99999	32.71562	**10.33034**	INVALID	84.99999
Kottenpark2009 - Netherlands	38	93	53	48		1301	31.99999	33.99999	**25.1403**	337.99999	97.96060
Woodlands2009 - South	42	40		30		1353	0.00121	2.00047	2.00012	59.00336	**0.00094**
School - Spain	35	66	4	21		439	0.04005	0.01247	**0.00597**	63.13873	0.01927
WesternGreeceUni3 - Greece	35	19		6		210	0.00016	0.00010	**0.00005**	14.00198	30.00002
WesternGreeceUni4 - Greece	35	19		12		262	0.00030	0.00016	**0.00005**	233.00277	35.00070
WesternGreeceUni5 - Greece	35	18		6		184	0.00004	0.00001	**0.00000**	9.00174	4.00013
						Average ranking	2.56	2.69	**1.36**	4.64	2.91

5.2.2 Comparison to the CHeSC 2011 hyper-heuristics

In CHeSC 2011, the competing hyper-heuristics are run for thirty one trials on the reference machine and the median result is used for comparison of the approaches based on the Formula1 system. The 20 submitted hyper-heuristics competed over thirty problem instances, five coming from each of the six problem domains: boolean satisfiability (SAT), one dimensional bin packing (BP), permutation flow shop (PFS) and personnel scheduling (PS), travelling salesman problem (TSP) and vehicle routing problem (VRP). The maximum overall score that a hyper-heuristic could achieve was 300. The performance of our hyper-heuristic is compared to all 20 competition entries based on the Formula1 scoring system. Table 5 summarises the overall results. Greedy gradient−simulated annealing ranks the tenth among others with a total score of 54.0. The proposed hyper-heuristic delivers the worst performance on the vehicle routing and traveling salesman problem domains.

Di Gaspero and Urli (2012) suggested the use of normalised objective function values (cost) as a measurement to rank the hyper-heuristics for a given domain. The median results obtained from the algorithms from each domain are normalised to a value in [0,1] based on the maximum and minimum cost obtained for all instances. Hence, the box plots for the algorithms in a given domain would indicate the relative variation of each competing hyper-heuristic for that problem domain. Figure 6 illustrates the ranking based on the median of the normalised cost for each problem domain. Lower the ranking, better an algorithm is. The greedy-gradient hyper-heuristic is the top in personnel scheduling problem domain when compared to the previously proposed CHeSC 2011 approaches. On the permutation flow shop domain, the greedy gradient−simulated annealing hyper-heuristic produces high quality solutions when compared to the other approaches and becomes the fourth bests performing hyper-heuristic. However, the proposed hyper-heuristic yields a poor performance on the other problem domains, particularly on the vehicle routing problem domain.

Percentage utilisation is the ratio of number of a given low level heuristic is invoked to the number of overall heuristic invocations. Figure 7 shows the average percentage utilisation over 10 runs for each low level heuristic considering the invocations in which an improvement is obtained while solving an arbitrarily chosen instance from each problem domain. Some low level heuristics do not make generate improvement in the quality of a solution. For example, in SAT, MU0, MU1 and HC1 are the only heuristics that generate improvements. However, the other heuristics may still be useful when combined with another heuristic, considering that worsening solutions could be accepted. Mutational heuristics dominate the hill climbing heuristics in improvement for SAT, while the situation is vice verse in all other problem domains. A similar phenomena is observed on the other instances as well.

Figure 8 shows the behaviour of the greedy gradient hyper-heuristic for an arbitrarily selected instance from each problem over 10 runs. In most of the cases, the approach rapidly improves the quality of the solution in hand. After a while, the improvement process

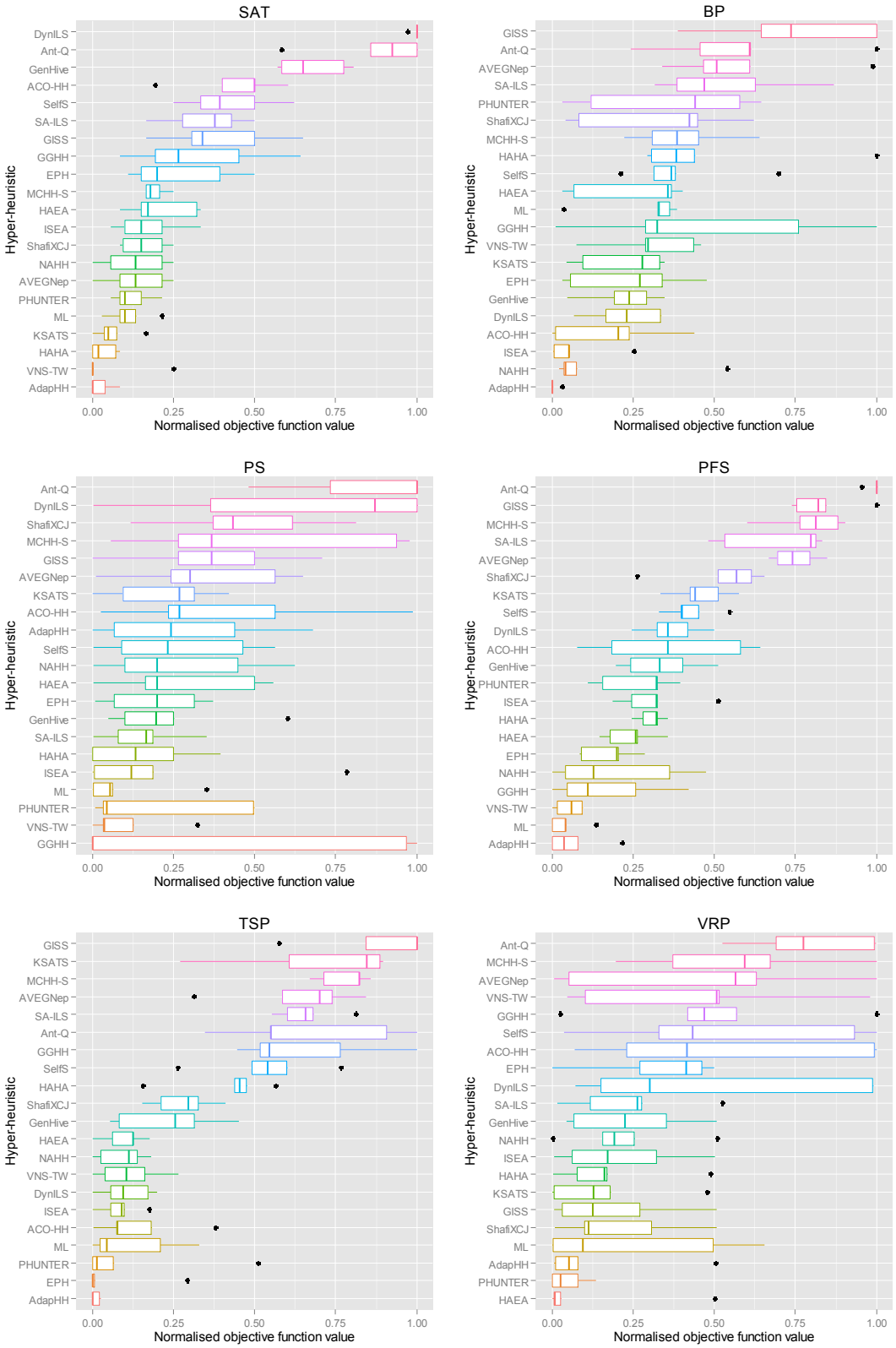

Fig. 6 Performance comparison (ranking) of hyper-heuristics for each CHeSC 2011 problem domain based on the results converted to the normalised objective function values. The dots in the box-plots are outliers.

Table 5 Comparisons of the different hyper-heuristics based on Formula1 scoring system.

HH	SAT	BP	PS	PFS	TSP	VRP	TOT
AdapHH	34.3	45	9	36	40.3	15	179.5
VNS-TW	34.3	2	35.5	31	17.3	6	126
ML	14	11	27.5	38	13	22	125.5
PHUNTER	10	3	11.5	7.5	26.3	33	91.3
EPH	0	10	8.5	19	36.3	12	85.8
NAHH	14	19	1	22	12	6	74
HAHA	32.3	0	23.5	0.8	0	14	70.6
ISEA	6	29	13.5	1.5	12	5	67
KSATS-HH	23.5	9	7.5	0	0	22	62
GGHH	**4**	**9**	**23**	**18**	**0**	**0**	**54**
HAEA	0.5	3	1	6.8	11	27	49.3
ACO-HH	0	20	0	8.3	8	2	38.3
GenHive	0	12	5.5	6	3	6	32.5
DynILS	0	12	0	0	13	1	26
XCJ	5.5	11	0	0	0	5	21.5
AVEG-Nep	12	0	0	0	0	9	21
SA-ILS	0.3	0	16	0	0	4	20.3
GISS	0.3	0	10	0	0	6	16.3
SelfSearch	0	0	2	0	3	0	5
MCHH-S	4.3	0	0	0	0	0	4.3
Ant-Q	0	0	0	0	0	0	0

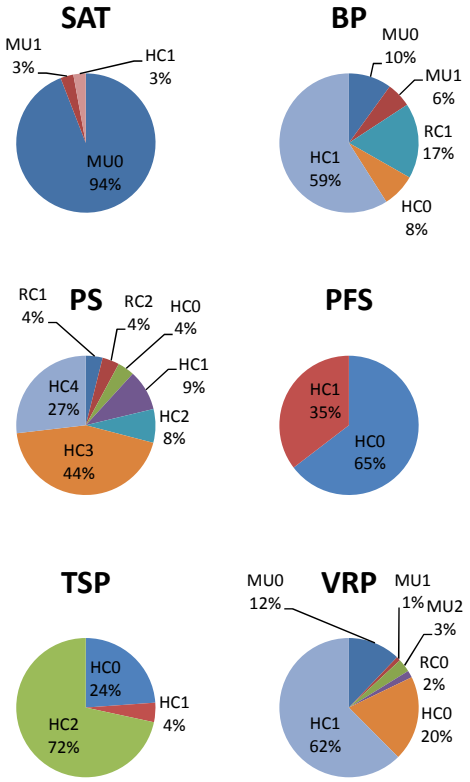

Fig. 7 Average percentage utilisation of low level heuristics over 10 runs while solving an arbitrary instance from each problem domain.

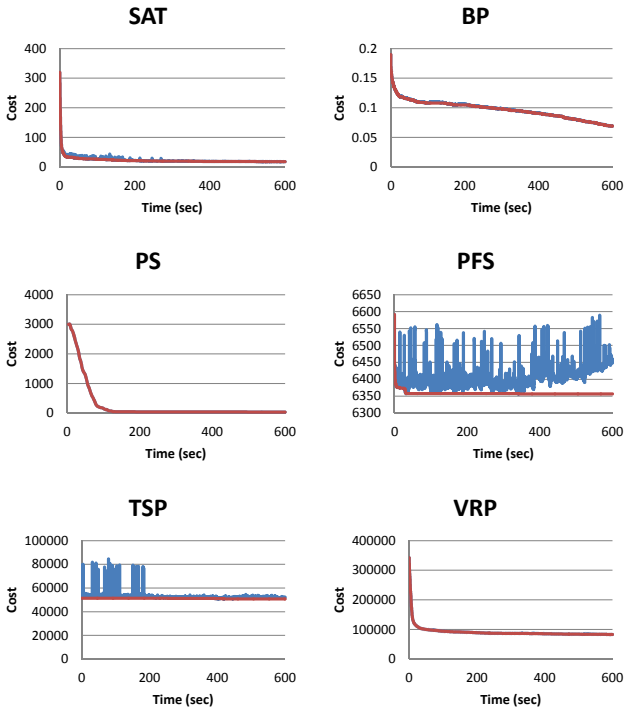

Fig. 8 Plots of the cost versus time over 10 runs while solving an arbitrary instance from each problem domain. Blue curve for the "average cost" and red curve for the "average best cost".

slows down as the approach reaches a local optimum. A similar phenomena is observed for almost all other instances from each problem domain.

6 Conclusion

A goal of hyper-heuristic research is to raise the level of generality by providing automated methodologies which are applicable to a variety of problem domains. In this

study, we have introduced a greedy-gradient−simulated annealing selection hyper-heuristic which automates the process of mixing perturbative heuristics. A set of selection hyper-heuristics using four different heuristic selection methods, including the proposed method are tested on a real world problem obtained from the Computer Engineering Department at Yeditepe University and eight problem instances which are randomly generated based on the definition of the given problem. The results show that the proposed greedy gradient heuristic selection method when combined with simulated annealing acceptance criterion outperforms the other selection hyper-heuristics. Although the performance of the new hyper-heuristic is evaluated on a new problem, it is our intention to extend our studies and investigate its performance across the other university course timetabling instances, such as ITC2007. The proposed methodology was particularly designed for a software tool implementing a user interface and the hyper-heuristic framework including all low level heuristics to deal with a curriculum-based university course timetabling problem at Yeditepe University. To test the level of generality that the proposed hyper-heuristic achieves, it is applied to the high school timetabling problem and six other problem domains obtained from a hyper-heuristic benchmark and compared to the previously proposed selection hyper-heuristics. The results show that our hyper-heuristic is a viable general methodology performing extremely well on not only course timetabling but also particularly personnel scheduling domain as well. Moreover, the proposed hyper-heuristic became the second best approach among the competing algorithms for high school timetabling. Although we have not performed any extensive parameter tuning on our hyper-heuristic, still the parameter that simulated annealing introduces can be considered as its weakness. Özcan et al (2008) observed that the acceptance criteria could make significant impact on the performance of the hyper-heuristics. As future work, we would like to analyse the effect of using other move acceptance criteria preferably requiring no parameter tuning in combination with the greedy gradient heuristic selection method. Additionally, crossover operators provided in each hyper-heuristic benchmark problem domain are ignored by our hyper-heuristic during the experiments, since crossover requires two solutions as input necessitating another top level mechanism to decide on those solutions. We plan to modify our hyper-heuristic to handle such operations and investigate into the benefit of using crossover. Finally, we would like to improve the performance of our selection hyper-heuristic further by incorporating the dominance-based method as described in Özcan and Kheiri (2012) which aims to reduce the set of the low level heuristics automatically during the search process.

References

Abramson D (1991) Constructing school timetables using simulated annealing: Sequential and parallel algorithms. Management Science 37(1):98–113

Abramson D, Dang H, Krisnamoorthy M (1999) Simulated annealing cooling schedules for the school timetabling problem. Asia-Pacific Journal of Operational Research 16:1–22

Alkan A, Özcan E (2003) Memetic algorithms for timetabling. In: Congress on Evolutionary Computation, CEC '03., vol 3, pp 1796 – 1802

Bai R, Kendall G (2005) An investigation of automated planograms using a simulated annealing based hyper-heuristics. In: Ibaraki T, Nonobe K, Yagiura M (eds) Metaheuristics: Progress as Real Problem Solver, Springer, pp 87–108

Bai R, Burke E, Gendreau M, Kendall G, McCollum B (2007a) Memory length in hyper-heuristics: An empirical study. In: IEEE Symposium on Computational Intelligence in Scheduling, SCIS '07., pp 173 –178

Bai R, Burke EK, Kendall G, McCollum B (2007b) 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 (2007) An experimental study on hyper-heuristics and exam timetabling. In: Burke E, Rudov H (eds) Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science, vol 3867, Springer Berlin / Heidelberg, pp 394–412

Burke E, Kendall G, Mısır M, Özcan E (2012) Monte carlo hyper-heuristics for examination timetabling. Annals of Operations Research 196(1):73–90

Burke EK, Kendall G, Soubeiga E (2003) A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9(6):451–470

Burke EK, Petrovic S, Qu R (2006) Case-based heuristic selection for timetabling problems. Journal of Scheduling 9(2):115–132

Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E, Woodward JR (2010) A classification of hyper-heuristics approaches. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol 57, 2nd edn, Springer, chap 15, pp 449–468

Burke EK, 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 DOI 10.1057/jors.2013.71

Colorni A, Dorigo M, Maniezzo V (1992) A genetic algorithm to solve the timetable problem. Tech. Rep. 90-060, Politecnico di Milano, Italy

Cowling P, Kendall G, Soubeiga E (2001) A hyperheuristic approach to scheduling a sales summit. In: Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling, Springer-Verlag, London, UK, pp 176–190

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

Di Gaspero L, Urli T (2012) Evaluation of a family of reinforcement learning cross-domain optimization heuristics. In: Hamadi Y, Schoenauer M (eds) Learning and Intelligent Optimization, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, pp 384–389

Domrös J, Homberger J (2012) An evolutionary algorithm for high school timetabling. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp 485–488

Erben W, Keppler J (1996) A genetic algorithm solving a weekly course-timetabling problem. In: Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, Springer-Verlag, London, UK, pp 198–211

Even S, Itai A, Shamir A (1976) On the Complexity of Timetable and Multicommodity Flow Problems. SIAM Journal on Computing 5(4):691–703

Filho GR, Antonio L, Lorena LAN (2001) A constructive evolutionary approach to school timetabling. In: Proceedings of the EvoWorkshops on Applications of Evolutionary Computing, Springer-Verlag, London, UK, pp 130–139

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

Fonseca GHG, Santos HG, Toffolo TAM, Brito SS, Souza MJF (2012) A sa-ils approach for the high school timetabling problem. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp 493–496

Hertz A (1992) Finding a feasible course schedule using tabu search. Discrete Applied Mathematics 35(3):255 – 270

Kalender M, Kheiri A, Özcan E, Burke E (2012) A greedy gradient-simulated annealing hyper-heuristic for a curriculum-based course timetabling problem. In: 2012 12th UK Workshop on Computational Intelligence, UKCI 2012

Kheiri A, Özcan E, Parkes AJ (2012) Hysst: Hyper-heuristic search strategies and timetabling. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp 497–499

Lehre P, Özcan E (2013) A runtime analysis of simple hyper-heuristics: To mix or not to mix operators. In: FOGA 2013 - Proceedings of the 12th ACM Workshop on Foundations of Genetic Algorithms, pp 97–104

Lewis R (2007) A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30(1):167–190

Lewis R, Paechter B, Rossi-Doria O (2007) Metaheuristics for university course timetabling. In: Dahal K, Tan K, Cowling P (eds) Evolutionary Scheduling (Studies in Computational Intelligence vol. 49), Springer-Verlag, Berlin, pp 237–272

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

Nareyek A (2004) Choosing search heuristics by non-stationary reinforcement learning. In: Resende MGC, de Sousa JP, Viana A (eds) Metaheuristics: Computer Desicion-Making, Kluwer Academic Publishers, Norwell, MA, USA, pp 523–544

Ochoa G, Hyde M, Curtois T, Vazquez-Rodriguez J, Walker J, Gendreau M, Kendall G, McCollum B, Parkes A, Petrovic S, Burke E (2012) Hyflex: A benchmark framework for cross-domain heuristic search. In: Hao JK, Middendorf M (eds) European Conference on Evolutionary Computation in Combinatorial Optimisation, EvoCOP '12., Springer, Heidelberg, LNCS, vol 7245, pp 136–147

Özcan E, Ersoy E (2005) Final exam scheduler - fes. In: The 2005 IEEE Congress on Evolutionary Computation., vol 2, pp 1356–1363

Özcan E, Kheiri A (2012) A hyper-heuristic based on random gradient, greedy and dominance. In: Gelenbe E, Lent R, Sakellari G (eds) Computer and Information Sciences II, Springer London, pp 557–563

Özcan E, Bilgin B, Korkmaz EE (2006) Hill climbers and mutational heuristics in hyperheuristics. In: Runarsson TP, Beyer HG, Burke E, Merelo-Guerv´s JJ, Whitley LD, Yao X (eds) Parallel Problem Solv-

ing from Nature - PPSN IX, Lecture Notes in Computer Science, vol 4193, Springer Berlin Heidelberg, pp 202–211

Özcan E, Bilgin B, Korkmaz EE (2008) A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12(1):3–23

Özcan E, Parkes AJ, Alkan A (2012) The interleaved constructive memetic algorithm and its application to timetabling. Computers & Operations Research 39(10):2310 – 2322

Paechter B, Rankin R, Cumming A, Fogarty T (1998) Timetabling the classes of an entire university with an evolutionary algorithm. In: Eiben A, Bäck T, Schoenauer M, Schwefel HP (eds) Parallel Problem Solving from Nature PPSN V, Lecture Notes in Computer Science, vol 1498, Springer Berlin Heidelberg, pp 865–874

Post G, Gaspero LD, Kingston JH, McCollum B, Schaerf A (2012) The third international timetabling competition. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp 479–484

Schaerf A (1996) Tabu search techniques for large high-school timetabling problems. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI '96., AAAI Press, pp 363–368

Socha K, Knowles J, Sampels M (2002) A max-min ant system for the university course timetabling problem. In: Proceedings of the Third International Workshop on Ant Algorithms, ANTS '02., Springer-Verlag, London, UK, pp 1–13

Sørensen M, Kristiansen S, Stidsen TR (2012) International timetabling competition 2011: an adaptive large neighborhood search algorithm. In: Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), pp 489–492

Swan J, Özcan E, Kendall G (2011) Hyperion - a recursive hyper-heuristic framework. In: Coello CAC (ed) LION, Springer, Lecture Notes in Computer Science, vol 6683, pp 616–630

de Werra D (1997) The combinatorics of timetabling. European Journal of Operational Research 96(3):504 – 513