A Greedy Gradient-Simulated Annealing Hyper-heuristic for a Curriculum-based Course Timetabling Problem

Murat Kalender*, Ahmed Kheiri†, Ender Özcan†, and Edmund K. Burke‡
*Yeditepe University, Computer Engineering Department
Inonu Mh., Kayisdagi Cd., 34755 Kadikoy/Istanbul, Turkey
mkalendertr@yahoo.com
†University of Nottingham, School of Computer Science
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
axk, exo@cs.nott.ac.uk
‡University of Stirling
Cottrell Building, Stirling FK9 4LA, UK
e.k.burke@stir.ac.uk

***Abstract*—The course timetabling problem is a well known constraint optimization problem which has been of interest to researchers as well as practitioners. Due to the NP-hard nature of the problem, the traditional exact approaches might fail to find a solution even for a given instance. Hyper-heuristics which search the space of heuristics for high quality solutions are alternative methods that have been increasingly used in solving such problems. In this study, a curriculum based course timetabling problem at Yeditepe University is described. An improvement oriented heuristic selection strategy combined with a simulated annealing move acceptance as a hyper-heuristic utilizing a set of low level constraint oriented neighbourhood heuristics is investigated for solving this problem. The proposed hyper-heuristic was initially developed to handle a variety of problems in a particular domain with different properties considering the nature of the low level heuristics. On the other hand, a goal of hyper-heuristic development is to build methods which are general. Hence, the proposed hyper-heuristic is applied to six other problem domains and its performance is compared to different state-of-the-art hyper-heuristics to test its level of generality. The empirical results show that the proposed method is sufficiently general and powerful.**

I. Introduction

A. Selection Hyper-heuristics

A *hyper-heuristic* is a high level (meta-)heuristic which performs a search over the space of (meta-)heuristics for solving computationally hard problems [1], [2]. A hyper-heuristic processes problem independent information obtained from the low level heuristics which operate at the problem level. The hyper-heuristic methodologies are used to *select* or *generate* heuristics [3]. This study focuses on a selection hyper-heuristic framework which contains two essential components: *heuristic selection* and *move acceptance* methods, as identified in [4]. Such a hyper-heuristic attempts to improve a solution (or solutions) by selecting a perturbative heuristic (or heuristics) followed by a decision to accept or reject the resulting solution(s) at a given step during the search process. A selection hyper-heuristic of this sort will be denoted as *heuristic selection−move acceptance* from this point onward.

Cowling et al. [2] experimented with selection hyper-heuristics that combine *random descent*, *random permutation*, *random permutation descent*, *greedy* (GR) and *choice function* (CF) heuristic selection mechanisms by *accept all moves* (AM) and *accept only improving moves* as move acceptance strategies. Greedy applies all low level heuristics to the current solution, generating multiple new solutions and chooses the one that achieves the best objective value. Choice function utilises a mechanism that scores each low level heuristic based on its individual performance, pair-wise successive performance and the duration since the last time a heuristic was invoked. The heuristic having the maximum score is chosen and applied to the candidate solution at each step. Both GR and CF are online learning mechanisms, since they both get and use feedback during the search process. GR has a very short term memory as the feedback (i.e., best candidate solution among all) is used instantaneously and then forgotten [1]. At the end of the experiments, the authors observed that CF−AM was the most promising hyper-heuristic [2]. The performance strength of CF has been illustrated in the other studies as well [4], [5], [6].

A variety of move acceptance methods have been investigated as a selection hyper-heuristic component, including simple deterministic methods, such as accept all moves, only improving moves, equal and improving moves, and more elaborate methods, such as great deluge and simulated annealing [1]. Simulated annealing accepts improving moves and 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 [7], Bai et al. [8] and Bilgin et al. [6] reported the success of simulated annealing as a move acceptance on the shelf allocation and examination timetabling problems, respectively. Moreover, Bilgin et al. [6] 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 CF−SA.

B. University Course Timetabling Problem

The educational timetabling problems, such as university course timetabling, examination timetabling and school timetabling, are NP-hard real-world constraint optimisation problems [9], [10]. University course timetabling is the focus of this study, which requires a search for the best course schedule and the best allocation of limited resources in an educational institution subject to a set of constraints. There are two basic constraint types in timetabling problems: $hard$, and $soft$. The hard constraints require absolute compliance, whereas the soft constraints characterise preferences. Types (and categories) of the constraints, number of the constraints, courses to be scheduled and the resources in a problem might influence its difficulty level. As a subclass, university course timetabling problems have been studied by many researchers [11], [12], [13], [14]. The university course timetabling problems can be further subdivided into (i) post-enrollment problems in which the student enrollment is known, and (ii) curriculum based problems in which the student enrollment is not known and student curriculums are available prior to the timetabling process [15].

Due to the nature of timetabling problems (e.g., unstructured search space, immense size of the search landscape, constraints etc.), meta-heuristics are preferred in most of the previous studies. Genetic algorithm (GA), memetic algorithm (MA), ant colony optimization, simulated annealing (SA) and tabu search (TS) are the most common meta-heuristics used for solving different types of course timetabling problems across different institutions.

Abramson [16] utilized simulated annealing for course timetabling. Colorni et. al. [17] evaluated a variety of meta-heuristics based on GA, SA and TS on some Italian high-school data. They observed that MA combining GA and local search performed better. Herz [18] employed TS for obtaining the best course schedules. Erben et al. [13] generated a weekly timetable for a heavily constraint problem instance using GAs with smart operators. They used binary encoding as a representation scheme. Schaerf [19] proposed an interactive interface for timetabling and embedded TS as a search tool for solving some high-school course timetabling problems. Paechter et al. [20] also used an interactive tool that allowed users to visualize violated objectives and modify the objectives during a run for solving Napier University timetabling problem. They used an evolutionary algorithm within their tool for performing the search. Abramson et al. [21] tested different cooling schedules within SA for course timetabling. Filho et al. [22] formulated timetabling problem as a clustering problem and applied a constructive GA for solving timetabling problems of public schools in Brazil. Socha et al. [14] 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 hybridized a violation directed hierarchical hill climbing method (VDHC) using constraint oriented neighbourhood heuristics in [23] 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 [24] for solving a variant of a course timetabling problem. Burke et al. [25] 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. [26] 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. [25] used tabu search hyper-heuristic to build solutions using graph coloring heuristics for solving timetabling problems. Detecting the state of the art method for the university course timetabling among modern approaches was one of the deriving ideas behind the International Timetabling Competition ITC2007 hosted by PATAT and WATT[1]. The winner of the Curriculum based Course Timetabling track was a hybrid approach designed by Müller [27].

In this study, we introduce a curriculum based university course timetabling problem constantly dealt with at Yeditepe University, Faculty of Engineering and Architecture, Computer Engineering Department and hyper-heuristic solution to the problem, which combines a fast reacting greedy and gradient heuristic selection mechanism with simulated annealing. A goal of hyper-heuristic research is to raise the level of generality by providing methodologies which are applicable to a variety of problem domains.

There are tools like Hyperion [28] and Hyflex [29] which are available for rapid development and research of hyper-heuristics or meta-heuristics. Hyperion provides a general recursive framework that supports the selection hyper-heuristic frameworks provided in [4]. Hyflex provides an object-oriented hyper-heuristic framework, having a support for the problem domains of boolean satisfiability, one-dimensional bin-packing, permutation flow-shop, personnel scheduling, travelling salesman problem (TSP) and vehicle routing problem (VRP). In this study, Hyflex is used to test the generality of the proposed hyper-heuristic across the different problem domains. Hyflex implementation supports the idea that no domain specific information is passed through the domain barrier from the domain level to the hyper-heuristic level. On the other hand, there are different views on what constitutes a domain specific information and generality of a hyper-heuristic method. The proposed hyper-heuristic similar to the state-of-the-art hyper-heuristic approach [30] does not discriminate

[1] http://www.cs.qub.ac.uk/itc2007/

between the types of the hyper-heuristics and does not use the type information for the low level heuristics in a given problem domain. Even if one decides to make use of this information say to create a generalised iterated local search framework where a prefixed hill climbing is applied after a perturbative low level heuristic. It is difficult to do this with the current version of Hyflex problem domain implementations, since the annotation of low level heuristics does not provide full information. For example, a ruin and create heuristic could be a mutational or hill climbing heuristic, which goes for the crossover operators as well.

Section II describes the greedy gradient hyper-heuristic selection methodology used for solving a curriculum-based course timetabling problem. It provides the constraints of the problem and the components of the proposed selection hyper-heuristic, including heuristic selection, move acceptance and low level heuristics. Section III provides the empirical results. Finally, Section IV presents the conclusions.

II. Methodology

A. Greedy Gradient Heuristic Selection

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 [31], [32]. 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 fitness 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 (Figure 1, steps 2, 3). Each heuristic is called one by one using the same solution at hand and the fitness 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 (Figure 1, step 4). In this study, max function is used, choosing the heuristic that generates the best improvement. After applying the selected heuristic, its score is updated right away using the fitness change. This strategy neither makes use of a periodic update of scores as in [8], nor forgets the scores as soon as a heuristic is selected as in a greedy method [2]. 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.

B. Curriculum-based Course Timetabling Problem

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 corequisites. The timetables are produced for the regular students. A course consists of lecture, 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.

```
1. Greedy_Gradient_Choose_Heuristic(scores, current solution)
2.     if (all heuristic scores are 0) then
3.         invoke each heuristic using the current solution and
           record fitness change as its score // 0 for worsening
4.     choose a heuristic based on the scores
       in case of a tie, use a tie breaking strategy
5.     return (chosen heuristic id for invocation)
```

Fig. 1. Pseudocode of the greedy gradient heuristic selection method

- **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 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.

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

Fig. 2. Yeditepe University course timetabling template.

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.

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 low level heuristics are designed as a set of constraint based neighbourhood operators as in [23]. Each low level heuristic attempts to improve upon a corresponding constraint. A course is randomly rescheduled to a constraint based neighbourhood. 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. The evaluation function 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:

$$evaluationFunction(T) = \sum_{\forall i} w_i g_i(T) \qquad (2)$$

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 minimizes the evaluation function and the optimum value is zero indicating that there are no constraint violations. The weight values used during the experiments are provided in Table I.

III. Empirical Results

A. Hyper-heuristics for the Course Timetabling Problem

The performance of four different hyper-heuristics are investigated over eight randomly generated instances (rp1-8) and a real instance (cse) during the experiments. The experiments were performed on a PC P4 Processor 3 GHz, 512 RAM. A

TABLE I
CONFLICT WEIGHT VALUES OF THE HARD AND SOFT CONSTRAINTS

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

run terminates after solution found or time limit reached 600 seconds. Table II summarises the performance of each hyper-heuristic based on 50 runs for each instance. The rankings of the different hyper-heuristics in Table II are calculated according to the success ratios (the ratio of successful runs in which the optimal solution is found to the total number of runs), the average best fitness values and the average best duration values of the tests.

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 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 is to be able to solve the university timetabling problem for the whole university. The results show that GG−SA is promising in this respect.

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 optimal solution. This behaviour is illustrated in Figure 3 for the GG−SA and CF−SA on the rp8 instance.

TABLE II
PERFORMANCE RANKING OF EACH HYPER-HEURISTIC OVER A SET OF PROBLEM INSTANCES

label	events	lecturers	GG-SA	CF-SA	SR-SA	GR-SA
rp1	200	64	2.5	2.5	2.5	2.5
rp2	200	64	2	2	2	4
rp3	400	128	2	2	2	4
rp4	400	128	1	2	3	4
rp5	800	256	2	2	2	4
rp6	800	256	1	2	3	4
rp7	1600	512	1	2	3	4
rp8	1600	512	1	2	3	4
cse	200	64	2.5	2.5	2.5	2.5
		avr	1.67	2.11	2.56	3.67

B. Testing the Level of Generality

HyFlex (Hyper-heuristics Flexible framework) [29] is an object-oriented framework which has been developed recently to provide a software tool for the implementation and comparison of different hyper-heuristics. HyFlex has been used for CHeSC: Cross-Domain Heuristic Search Competition [2].

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

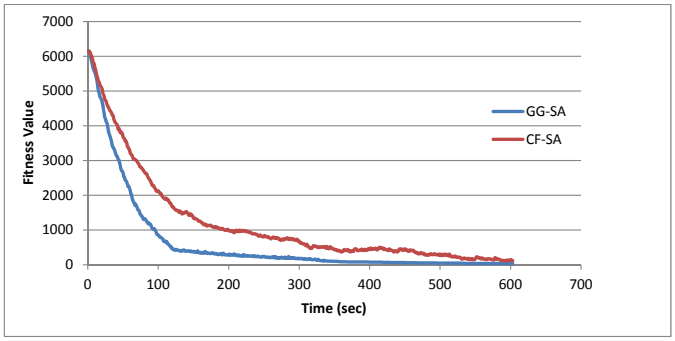

Fig. 3. Fitness of the current solution versus time for the top two hyper-heuristics on rp8

TABLE III
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
MAX-SAT	5	2	1	2	10
Bin Packing	3	2	2	1	8
Permutation Flow Shop	1	5	3	3	12
Personnel Scheduling	5	4	2	4	15
Travelling Salesman	5	3	1	4	13
Vehicle Routing	3	3	2	2	10

The best selection hyper-heuristic will be determined among CHeSC competitors which generalises the best across a set of five problem instances from six different problem domains.

HyFlex currently provides an implementation of six minimisation problems including Boolean Satisfiability (MAX-SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) each with different instances and a set of low-level heuristics. These low level heuristics are classified as mutational (MU), hill climbing (HC), ruin and re-create (RC), and crossover (XO) heuristics. The number of the low level heuristics for each heuristic/operator type for each problem domain is presented in Table III. Currently, there are 12 different instances for the first four problem domains and 10 for the last two problem domains. What is left is to design and implement a high-level strategy (hyper-heuristic) that intelligently select and apply suitable low-level heuristics from the set provided to each instant from the given domain to get the minimum objective function value in ten minutes. The crossover low level heuristics take two solutions as a parameter, combine them and return a new solution. In our approach we are working on a single point which leads us to exclude all the crossover heuristics

The proposed hyper-heuristic in this study is implemented as an extension to HyFlex. In the simulated annealing acceptance method, the value of ΔF is set to 0.01 of the objective value 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. The performance of the proposed hyper-heuristic is compared to the performances of eight different hyper-heuristics (HH1HH8) across four problem domains, each with 10 different instances, as provided at (www.asap.cs.nott.ac.uk/external/chesc2011/defaulthh.html) and in [33]. The comparison is based on the Formula1 scoring system. The best hyper-heuristic gets 10 points, the second gets 8, and then 6,5,4,3,2,1 and then all the remaining get zero point. These points are accumulated as a score for a hyper-heuristic over all instances from four problem domains each with ten instances. The problem domains are Boolean Satisfiability (MAX-SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS) and Personnel Scheduling (PS).

Table IV, V, VI and VII compare the performance of our proposed hyper-heuristic (PHH) to the others (HH1–HH8) over a set of problem instances for MAX-SAT, 1D bin packing, personnel scheduling and permutation flow shop, respectively, based on the objective values obtained at the end of each run.

In the MAX-SAT problem domain, our hyper-heuristic produces the best result in 3 out of 10 instances and there is a tie in 1 instance. It is the second hyper-heuristic in this domain. In the bin packing problem domain, our hyper-heuristic performs still well and produces the best result in 2 instances, but in the personnel scheduling problem, its performance is not as good as expected. In permutation flow shop, the proposed hyper-heuristic produces the best result in 2 instances. Table VIII provides the results for each problem domain. The proposed hyper-heuristic ranks the third with a total score of 211.5.

The performances of the proposed hyper-heuristic and twenty other hyper-heuristics which were the CHESC finalists are compared to each other. Five instances from each of the six problem domains are selected randomly for comparison. The solutions obtained from twenty one hyper-heuristics across all instances are put into the competition based on the Formula1 scoring system. Table IX summarises the results. The proposed hyper-heuristic ranks the tenth among others with a total score of 54.0.

TABLE IX
COMPARISONS OF THE DIFFERENT HYPER-HEURISTICS.

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
GG	**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

IV. CONCLUSION

In this study, a university course timetabling problem at Yeditepe University, Faculty of Engineering and Architecture is introduced. Moreover, the performance of a greedy-gradient−simulated annealing hyper-heuristic is investigated as a solution methodology. The proposed hyper-heuristic aims to automate the process of selecting and combining perturbative heuristics to solve the NP-hard curriculum based course timetabling problem. 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, Faculty of Engineering and Architecture 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 outperforms the other approaches when combined with simulated annealing acceptance criterion. 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. To test its level of generality, the proposed hyper-heuristic has been tested across different problem domains and compared to different previously proposed hyper-heuristics. The results show that the hyper-heuristic is a general methodology.

REFERENCES

[1] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *JORS*, to appear.

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

[3] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristics approaches," in *Handbook of Metaheuristics*, 2nd ed., ser. International Series in Operations Research & Management Science, M. Gendreau and J.-Y. Potvin, Eds. Springer, 2010, vol. 57, ch. 15, pp. 449–468.

[4] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[5] E. Burke, G. Kendall, M. Misir, and E. Özcan, "Monte carlo hyper-heuristics for examination timetabling," *Annals of Operations Research*, pp. 1–18, 2010.

[6] B. Bilgin, E. Özcan, and E. Korkmaz, "An experimental study on hyper-heuristics and exam timetabling," in *Practice and Theory of Automated Timetabling VI*, ser. Lecture Notes in Computer Science, E. Burke and H. Rudov, Eds. Springer Berlin / Heidelberg, 2007, vol. 3867, pp. 394–412.

TABLE IV

MAX-SAT, OBJECTIVE FUNCTION VALUES OBTAINED BY THE EIGHT HYPER-HEURISTICS AND THE NEW DEVELOPED HYPER-HEURISTIC ON THE 10 INSTANCES. THE BEST VALUES FOR EACH INSTANCE ARE HIGHLIGHTED IN BOLD.

Instance	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	PHH
Inst1	46	33	14	28	119	**12**	56	40	16
Inst2	40	33	36	50	136	34	38	66	**28**
Inst3	32	24	28	47	116	29	35	53	**17**
Inst4	16	13	35	24	60	15	15	25	**12**
Inst5	**9**	10	45	37	70	33	**9**	36	33
Inst6	22	**17**	52	52	106	51	24	55	36
Inst7	6	6	8	12	18	9	**5**	15	9
Inst8	**6**	**6**	8	11	13	11	**6**	14	7
Inst9	8	**7**	11	16	21	12	9	19	11
Inst10	**211**	**211**	221	239	259	215	217	239	**211**

TABLE V

1D BIN PACKING, OBJECTIVE FUNCTION VALUES OBTAINED BY THE EIGHT HYPER-HEURISTICS AND THE NEW DEVELOPED HYPER-HEURISTIC ON THE 10 INSTANCES. THE BEST VALUES FOR EACH INSTANCE ARE HIGHLIGHTED IN BOLD.

Instance	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	PHH
Inst1	0.0174	0.0176	**0.0108**	0.0120	0.0541	0.0157	0.0217	0.0714	0.0115
Inst2	0.0163	0.0165	**0.0071**	0.0077	0.0501	0.0129	0.0214	0.0712	0.0115
Inst3	0.0238	0.0229	0.0247	0.0230	0.0283	0.0231	0.0236	0.0308	**0.0228**
Inst4	0.0248	0.0249	0.0266	0.0243	0.0308	0.0257	0.0255	0.0327	**0.0241**
Inst5	0.0064	0.0062	**0.0003**	0.0046	0.0151	0.0066	0.0073	0.0218	0.0046
Inst6	0.0040	0.0085	**0.0036**	**0.0036**	0.0187	0.0089	0.0090	0.0234	0.0038
Inst7	0.1145	0.1047	**0.0107**	0.0312	0.1715	0.0538	0.1386	0.1666	0.0431
Inst8	0.1337	0.1285	**0.0168**	0.0640	0.1719	0.0942	0.1506	0.1799	0.0691
Inst9	**0.0559**	0.0569	0.0562	0.0944	0.0927	0.0608	0.0582	0.1267	0.0658
Inst10	0.0135	**0.0128**	0.0190	0.0309	0.0344	0.0166	0.0139	0.0428	0.0287

TABLE VI

PERSONNEL SCHEDULING, OBJECTIVE FUNCTION VALUES OBTAINED BY THE EIGHT HYPER-HEURISTICS AND THE NEW DEVELOPED HYPER-HEURISTIC ON THE 10 INSTANCES. THE BEST VALUES FOR EACH INSTANCE ARE HIGHLIGHTED IN BOLD.

Instance	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	PHH
Inst1	3346	3321	3389	**3318**	3338	8017	3344	3342	3398
Inst2	2220	2315	2400	2275	2454	21008	**2095**	2893	2595
Inst3	390	400	495	375	400	905	355	**340**	3175
Inst4	23	17	32	19	**16**	80	22	**16**	36
Inst5	23	26	32	24	24	81	**19**	28	36
Inst6	**17**	**17**	32	24	22	81	28	38	30
Inst7	**1111**	1119	1231	1118	1113	35391	1211	1490	1417
Inst8	**2188**	2202	2205	2221	2288	46661	2275	3959	2459
Inst9	**3163**	3255	3465	3360	3354	46952	3414	6905	3519
Inst10	11486	**9706**	12505	12994	9771	105850	9807	17224	9835

TABLE VII

PERMUTATION FLOW SHOP, OBJECTIVE FUNCTION VALUES OBTAINED BY THE EIGHT HYPER-HEURISTICS AND THE NEW DEVELOPED HYPER-HEURISTIC ON THE 10 INSTANCES. THE BEST VALUES FOR EACH INSTANCE ARE HIGHLIGHTED IN BOLD.

Instance	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	PHH
Inst1	6365	6380	6399	6312	6393	**6297**	6375	6323	6370
Inst2	6327	6330	6337	6281	6328	**6253**	6335	6288	6297
Inst3	6401	6410	6401	**6339**	6418	**6339**	6407	6364	6388
Inst4	6388	6408	6366	**6327**	6373	6366	6371	6363	6369
Inst5	6461	6470	6438	**6392**	6483	6405	6478	6422	6453
Inst6	10540	10546	10506	**10499**	10547	10509	10546	10542	10512
Inst7	10976	10965	10965	**10923**	10980	**10923**	10965	10956	10959
Inst8	26483	26490	26538	26409	26506	26418	26512	26396	**26322**
Inst9	26979	26929	26978	26890	26913	26920	26960	**26800**	26850
Inst10	26755	26794	26833	26731	26755	26715	26811	26716	**26714**

TABLE VIII
COMPARISONS OF THE DIFFERENT HYPER-HEURISTICS OVER EACH DOMAIN BASED ON FORMULA1 SCORES

Domain	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	PHH
MAX-SAT	61.0	77.0	40.0	24.5	1.0	49.0	56.5	14.5	66.5
1D Bin Packing	50.0	52.0	76.0	61.0	10.0	43.0	31.0	1.0	66.0
Personnel Scheduling	70.0	66.5	29.0	55.5	58.0	1.0	58.0	33.0	19.0
Permutation Flow Shop	29.0	19.5	26.0	81.0	16.5	74.5	18.5	65.0	60.0
Overall	210.0	215.0	171.0	222.0	85.5	167.5	164.0	113.5	211.5

[7] R. Bai and G. Kendall, "An investigation of automated planograms using a simulated annealing based hyper-heuristics," in *Metaheuristics: Progress as Real Problem Solver*, T. Ibaraki, K. Nonobe, and M. Yagiura, Eds. Springer, 2005, pp. 87–108.

[8] R. Bai, E. K. Burke, G. Kendall, and B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support," School of CSiT, University of Nottingham, Tech. Rep. Technical Report NOTTCS-TR-2007-8, 2007.

[9] S. Even, A. Itai, and A. Shamir, "On the Complexity of Timetable and Multicommodity Flow Problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[10] D. de Werra, "The combinatorics of timetabling," *European Journal of Operational Research*, vol. 96, no. 3, pp. 504 – 513, 1997.

[11] R. Lewis, "A survey of metaheuristic-based techniques for university timetabling problems," *OR Spectrum*, vol. 30, no. 1, pp. 167–190, 2007.

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

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

[14] K. Socha, J. Knowles, and M. Sampels, "A max-min ant system for the university course timetabling problem," in *Proceedings of the Third International Workshop on Ant Algorithms*, ser. ANTS '02. London, UK, UK: Springer-Verlag, 2002, pp. 1–13.

[15] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition," *INFORMS J. on Computing*, vol. 22, no. 1, pp. 120–130, Jan. 2010.

[16] D. Abramson, "Constructing school timetables using simulated annealing: Sequential and parallel algorithms," *Management Science*, vol. 37, no. 1, pp. 98–113, 1991.

[17] A. Colorni, M. Dorigo, and V. Maniezzo, "A genetic algorithm to solve the timetable problem," Politecnico di Milano, Italy, Tech. Rep. Technical Report No. 90-060, 1992.

[18] A. Hertz, "Finding a feasible course schedule using tabu search," *Discrete Applied Mathematics*, vol. 35, no. 3, pp. 255 – 270, 1992.

[19] A. Schaerf, "Tabu search techniques for large high-school timetabling problems," in *Proceedings of the thirteenth national conference on Artificial intelligence - Volume 1*, ser. AAAI'96. AAAI Press, 1996, pp. 363–368.

[20] B. Paechter, R. C. Rankin, A. Cumming, and T. C. Fogarty, "Timetabling the classes of an entire university with an evolutionary algorithm," in *V, Springer, LNCS*. Springer-Verlag, 1998, pp. 865–874.

[21] D. Abramson, H. Dang, and M. Krisnamoorthy, "Simulated annealing cooling schedules for the school timetabling problem," *Asia-Pacific Journal of Operational Research*, vol. 16, pp. 1–22, 1999.

[22] G. R. Filho, L. Antonio, and L. A. N. Lorena, "A constructive evolutionary approach to school timetabling," in *In Applications of Evolutionary Computing*. Springer Lecture, 2001, pp. 130–139.

[23] A. Alkan and E. Özcan, "Memetic algorithms for timetabling," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, vol. 3, 2003, pp. 1796 – 1802.

[24] E. Özcan, A. J. Parkes, and A. Alkan, "The interleaved constructive memetic algorithm and its application to timetabling," *Computers & Operations Research*, vol. 39, no. 10, pp. 2310 – 2322, 2012.

[25] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[26] E. K. Burke, S. Petrovic, and R. Qu, "Case-based heuristic selection for timetabling problems," *J. of Scheduling*, vol. 9, no. 2, pp. 115–132, 2006.

[27] T. Muller, "Itc2007 solver description: a hybrid approach," *Annals of Operations Research*, vol. 172, pp. 429–446, 2009.

[28] J. Swan, E. Özcan, and G. Kendall, "Hyperion - a recursive hyper-heuristic framework." in *LION*, ser. Lecture Notes in Computer Science, C. A. C. Coello, Ed., vol. 6683. Springer, 2011, pp. 616–630.

[29] E. Burke, T. Curtois, M. Hyde, G. Ochoa, and J. A. Vazquez-Rodriguez, "HyFlex: A Benchmark Framework for Cross-domain Heuristic Search," *ArXiv e-prints*, Jul. 2011.

[30] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe, "A new hyper-heuristic implementation in HyFlex: a study on generality," in *Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory & Application,*, J. Fowler, G. Kendall, and B. McCollum, Eds., Aug. 2011, pp. 374–393. [Online]. Available: https://lirias.kuleuven.be/handle/123456789/313980

[31] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Metaheuristics: Computer Decision-Making*, M. G. C. Resende and J. P. de Sousa, Eds. Kluwer, 2003, ch. 9, pp. 523–544.

[32] R. Bai, E. Burke, M. Gendreau, G. Kendall, and B. McCollum, "Memory length in hyper-heuristics: An empirical study," in *Computational Intelligence in Scheduling, SCIS '07. IEEE Symposium on*, april 2007, pp. 173 –178.

[33] E. K. Burke, T. Curtois, M. R. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J. A. V. Rodríguez, and M. Gendreau, "Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8.