An Experimental Study on Hyper-heuristics and Exam Timetabling

Burak Bilgin, Ender Özcan, Emin Erkan Korkmaz

Artificial Intelligence Laboratory (ARTI)
Yeditepe University, Department of Computer Engineering,
34755 Kadıköy/İstanbul, Turkey
Email: {bbilgin|eozcan|ekorkmaz}@cse.yeditepe.edu.tr

Abstract. Hyper-heuristics are proposed as a higher level of abstraction as compared to the metaheuristics. Hyper-heuristic methods deploy a set of simple heuristics and use only nonproblem-specific data, such as, fitness change or heuristic execution time. A typical iteration of a hyper-heuristic algorithm consists of two phases: heuristic selection method and move acceptance. In this paper, heuristic selection mechanisms and move acceptance criteria in hyper-heuristics are analyzed in depth. Seven heuristic selection methods, and five acceptance criteria are implemented. The performance of each selection and acceptance mechanism pair is evaluated on fourteen well-known benchmark functions and twenty-one exam timetabling problem instances.

1 Introduction

The term hyper-heuristic refers to a recent approach used as a search methodology [2, 3, 5, 11, 21]. It is a higher level of abstraction than metaheuristic methods. Hyper-heuristics involve an iterative strategy that chooses a heuristic to apply to a candidate solution of the problem at hand, at each step. Cowling et al. discusses properties of hyper-heuristics in [11]. An iteration of a hyper-heuristic can be subdivided into two parts; heuristic selection and move acceptance. In the hyper-heuristic literature, several heuristic selection and acceptance mechanisms are used [2, 3, 5, 11, 21]. However, no comprehensive study exists that compare the performances of these different mechanisms in depth.

Timetabling problems are real world constraint optimization problems. Due to their NP complete nature [16], traditional approaches might fail to generate a solution to a timetabling problem instance. Timetabling problems require assignment of *time-slots* (periods) and possibly some other resources to a set of events, subject to a set of constraints. Numerous researchers deal with different types of timetabling problems based on different types of constraints utilizing variety of approaches. *Employee timetabling*, *course timetabling* and *examination timetabling* are the research fields that attract the most attention. In this paper, seven heuristic selection methods and five different acceptance criteria are analyzed in depth. Their performance is measured on well-known benchmark functions. Moreover, thirty-five hyper-heuristics

generated by coupling all heuristic selection methods and all acceptance criteria with each other, are evaluated on a set of twenty-one exam timetabling benchmark problem instances, including Carter's benchmark [10] and Ozcan's benchmark [25].

The remainder of this paper is organized as follows. In Section 2 background is provided including hyper-heuristics, benchmark functions and exam timetabling. Experimental settings and results for benchmarks are given in Section 3. Hyper-heuristic experiments on exam timetabling are presented in Section 4. Finally, conclusions are discussed in Section 5.

2 Preliminaries

2.1 Hyper-heuristics

Hyper-heuristic methods are described by Cowling et al. [11] as an alternative method to meta-heuristics. Metaheuristics are 'problem-specific' solution methods, which require knowledge and experience about the problem domain and properties. Metaheuristics are mostly developed for a particular problem and require fine tuning of parameters. Therefore, they can be developed and deployed only by experts who have the sufficient knowledge and experience on the problem domain and the meta-heuristic search method. Hyper-heuristics, on the other hand are developed to be general optimization methods, which can be applied to any optimization problem easily. Hyper-heuristics can be considered as black box systems, which take the problem instance and several low level heuristics as input and which can produce the result independent of the problem characteristics. In this concept, hyper-heuristics use only non problem-specific data provided by each low level heuristic in order to select and apply them to candidate solution [3, 5, 11].

The selection mechanisms in the hyper-heuristic methods were emphasized in the initial phases of the research period. Cowling et al. [11] proposed three types of low level heuristic selection mechanisms to be used in hyper-heuristics; which are *Simple, Greedy* and *Choice Function*. There are four types of *Simple* heuristic selection mechanisms. *Simple Random* mechanism chooses a low level heuristic at a time randomly. *Random Descent* mechanism chooses a low level heuristic randomly and applies it repeatedly as long as it produces improving results. *Random Permutation* mechanism creates an initial permutation of the low level heuristics and at each iteration applies the next low level heuristic in the permutation. *Random Permutation Descent* mechanism is the same as *Random Permutation* mechanism, except that it applies the low level heuristic in turn repeatedly as long as it produces improving results. *Greedy* method calls each low level heuristic at each iteration and chooses the one that produces the most improving solution. *Choice Function* is the most complex one. It analyzes both the performance of each low level heuristic and each pair of low level heuristics. This analysis is based on the improvement and execution time. This mechanism also considers the overall performance. It attempts to focus the search as long as the improvement rate is high and broadens the search if the improvement rate

is low. For each of these low level heuristic selection mechanisms two simple acceptance criteria are defined. These are *AM*, where all moves are accepted and *OI* where only improving moves are accepted [11].

Burke et al. [5] proposed a *Tabu-Search* heuristic selection method. This mechanism ranks low level heuristics. At the beginning of the run each heuristic starts the execution with the minimum ranking. Every time a heuristic produces an improving movement its rank is increased by a positive reinforcement rate. The rank of the heuristics cannot exceed a predetermined maximum value. Whenever a heuristic cannot make an improving move; its rank is decreased by a negative reinforcement learning rate. Similarly the rank of a heuristic cannot be decreased to a value less than a predetermined minimum value. In the case of worsening moves, the heuristic is also added to the tabu list. Another parameter is the tabu duration which sets the maximum number of iterations a low level heuristic can stay in the tabu list. The tabu list is emptied every time there is a change in the fitness of the candidate solution [5].

Burke et al. [8] introduce a simple generic hyper-heuristic which utilizes constructive heuristics (graph coloring heuristics) to tackle timetabling problems. A tabu-search algorithm chooses among permutations of constructive heuristics according to their ability to construct complete, feasible and low cost timetables. At each iteration of the algorithm, if the selected permutation produces a feasible timetable, a deepest descent algorithm is applied to the obtained timetable. Burke et al. used this hyper-heuristic method in exam and university course timetabling problem instances. The proposed method worked well on the related benchmark problem instances [8].

Burke et al. [9] proposed a case based heuristic selection approach. A knowledge discovery method is employed to find the problem instances and situations where a specific heuristic has a good performance. The proposed method also explores the similarities between the problem instance and the source cases, in order to predict the heuristic that will perform best. Burke et al. applied Case-Based Heuristic Selection Approach to the exam and university course timetabling [9].

Ayob and Kendall [2] emphasized the role of the acceptance criterion in the hyper-heuristic. They introduced the *Monte Carlo Hyper-heuristic* which has a more complex acceptance criterion than *AM* or *OI* criteria. In this criterion, all of the improving moves are accepted and the non-improving moves can be accepted based on a probabilistic framework. Ayob and Kendall defined three probabilistic approaches to accept the non-improving moves. First approach, named as *Linear Monte Carlo* (*LMC*), uses a negative linear ratio of the probability of acceptance to the fitness worsening. Second approach named as, *Exponential Monte Carlo* (*EMC*), uses a negative exponential ratio of the probability of acceptance to the fitness worsening. Third approach, named as *Exponential Monte Carlo with Counter* (*EMCQ*), is an improvement over *Exponential Monte Carlo*. Again, the probability of accepting worsening moves decreases as the time passes. However if no improvement can be achieved over a series of consecutive iterations then this probability starts increasing again. As the heuristic selection mechanism, they all use simple random mechanism [2].

Kendall and Mohamad [21] introduced another hyper-heuristic method which also focuses on acceptance criterion rather than selection method. They used the *Great Deluge Algorithm* as the acceptance criterion and S*imple Random* as heuristic selec-

tion method. In the *Great Deluge Algorithm* initial fitness is set as initial level. At each step, the moves which produce fitness values less than the level are accepted. At each step the level is also decreased by a factor [21].

Gaw et al. [17] presented a research on the choice function hyper-heuristics, generalized low-level heuristics, and utilization of parallel computing environments for hyper-heuristics. An abstract low level heuristic model is proposed which can be easily implemented to be a functional low level heuristic tackling a specific problem type. The choice function hyper-heuristic and the low-level heuristics are improved to evaluate a broader range of the data. Two types of distributed hyper-heuristic approaches are introduced. The first approach is a single hyper-heuristic, multiple low-level heuristics which are executed on different nodes and focus on different areas of the timetable. The second approach utilizes multiple hyper-heuristics each of which work on a different node. In this approach, hyper-heuristics collaborate during the execution [17].

According to this survey it is concluded that several heuristic selection methods and acceptance criteria are introduced for hyper-heuristics framework. Each pair of the heuristic selection and acceptance mechanism can be used as a different hyper-heuristic method. Despite this fact, such combinations have not been studied in the literature. In this study, seven heuristic selection mechanisms, which are Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Choice-Function, Tabu-Search, Greedy heuristic selection mechanisms, are implemented. For each heuristic selection method five acceptance criteria: *AM*, *OI*, *IE*, a *Great Deluge* and a *Monte Carlo* are used. As a result a broad range of hyper-heuristic variants are obtained. These variants are tested on mathematical objective functions and exam timetabling Problems.

2.2 Benchmark Functions

Well-defined problem sets are useful to measure the performance of optimization methods such as genetic algorithms, memetic algorithms and hyper-heuristics. Benchmark functions which are based on mathematical functions or bit strings can be used as objective functions to carry out such tests. The characteristics of these benchmark functions are explicit. The difficulty levels of most benchmark functions are adjustable by setting their parameters. In this study, fourteen different benchmark functions are chosen to evaluate the hyper-heuristics.

The benchmark functions presented in Table 1 are continuous functions, and *Royal Road Function*, *Goldberg's 3 bit Deceptive Function* [18], [19] and *Whitley's 4 bit Deceptive Function* [31] are discrete functions. Their deceptive nature is due to the large Hamming Distance between the global optimum and the local optima. To increase the difficulty of the problem *n* dimensions of these functions can be combined by a summation operator.

The candidate solutions to all the continuous functions are encoded as bit strings using gray code. The properties of the benchmark functions are presented in Table 1. The modality property indicates the number of optima in the search space (i.e. between bounds). Unimodal benchmark functions have a single optimum. Multimodal

benchmark functions contain more than one optimum in their search space. Such functions contain at least one additional local optimum in which a search method can get stuck.

Table 1. Properties of benchmark functions, *lb* indicates the lower bound, *ub* indicates the upper bound of the search space, *opt* indicates the global optimum in the search space

Function, [*Source*]	*lb*	*ub*	*opt*	*Continuity*	*Modality*
Sphere, [13]	-5.12	5.12	0	Continuous	Unimodal
Rosenbrock, [13]	-2.048	2.048	0	Continuous	Unimodal
Step, [13]	-5.12	5.12	0	Continuous	Unimodal
Quartic, [13]	-1.28	1.28	1	Continuous	Multimodal
Foxhole, [13]	-65.536	65.536	0	Continuous	Multimodal
Rastrigin, [28]	-5.12	5.12	0	Continuous	Multimodal
Schwefel, [29]	-500	500	0	Continuous	Multimodal
Griewangk, [19]	-600	600	0	Continuous	Multimodal
Ackley, [1]	-32.768	32.768	0	Continuous	Multimodal
Easom, [15]	-100	100	-1	Continuous	Unimodal
Rotated Hyperellipsoid,[13]	-65.536	65.536	0	Continuous	Unimodal
Royal Road, [23]	-	-	0	Discrete	-
Goldberg, [17, 18]	-	-	0	Discrete	-
Whitley, [30]	-	-	0	Discrete	-

2.3 Exam Timetabling

Burke et al. [4, 6] applied a light or a heavy mutation, randomly selecting one, followed by a hill climbing method. Investigation of various combinations of Constraint Satisfaction Strategies with GAs for solving exam timetabling problems can be found in [22]. Paquete et. al. [27] applied a multiobjective evolutionary algorithm (MOEA) based on pareto ranking for solving exam timetabling problem in the Unit of Exact and Human Sciences at University of Algarve. Two objectives were determined as to minimize the number of conflicts within the same *group* and the conflicts among different *groups*. Wong et. al. [32] used a GA utilizing a non-elitist replacement strategy to solve a single exam timetabling problem at École de Technologie Supérieure. After genetic operators were applied, violations were fixed in a hill climbing procedure.

Carter et. al. [10] applied different heuristic orderings based on graph coloring. Their experimental data became one of the commonly used exam timetabling benchmarks. Gaspero and Schaerf [14] analyzed tabu search approach using graph coloring based heuristics. Merlot et al. [23] explored a hybrid approach for solving the exam timetabling problem that produces an initial feasible timetable via constraint programming. The method, then applies simulated annealing with hill climbing to improve the solution. Petrovic et al. [28] introduced a case based reasoning system to create initial solutions to be used by great deluge algorithm. Burke et al. [7] proposed a general and fast adaptive method that arranges the heuristic to be used for ordering exams to be scheduled next. Their algorithm produced comparable results on a set of benchmark problems with the current state of the art. Ozcan and Ersoy [25] used a

violation directed adaptive hill climber within a memetic algorithm to solve exam timetabling problem. A Java tool named FES is introduced by Ozcan in [26] which utilizes XML as input/output format.

Exam timetabling problem can be formulated as a constraint optimization problem by a 3-tuple (*V, D, C*). *V* is a finite set of examinations, *D* is a finite set of domains of variables, and C is a finite set of constraints to be satisfied. In this representation a variable stands for an exam schedule of a course. Exam timetabling involves a search for a solution, where values from domains (timeslots) are assigned to all variables while satisfying all the constraints.

The set of constraints for exam timetabling problem differs from institution to institution. In this study, three constraints are defined and used as described in [25]:

(i) A student cannot be scheduled to two exams at the same time slot.

(ii) If a student is scheduled to two exams in the same day, these should not be assigned to consecutive timeslots.

(iii) The total capacity for a timeslot cannot be exceeded.

3 Hyper-heuristics for Benchmark Functions

3.1 Benchmark Function Heuristics

Six heuristics were implemented to be used with hyper-heuristics on benchmark functions. Half of these are hill-climbing methods and the remaining half are mutational operators combined with a hill climber.

Next Ascent Hill Climber makes number of bits times iterations at each heuristic call. Starting from the most significant bit, at each iteration it inverts the next bit in the bit string. If there is a fitness improvement, the modified candidate solution is accepted as the current candidate solution [24]. *Davis' Bit Hill Climber* is the same as Next Ascent Hill Climber but it does not modify the bit sequentially but in the sequence of a randomly determined permutation [12]. *Random Mutation Hill Climber* chooses a bit randomly and inverts it. Again the modified candidate solution becomes the current candidate solution, if the fitness is improved. This step is repeated for total number of bits in the candidate solution times at each heuristic call [24].

Mutational heuristics are *Swap Dimension*, *Dimensional Mutation* and *Hypermutation*. *Swap Dimension* heuristic randomly chooses two different dimensions in the candidate solution and swaps them. *Dimensional Mutation* heuristic randomly chooses a dimension and inverts each bit in this dimension with the probability 0.5. *Hypermutation* randomly inverts each bit in the candidate solution with the probability 0.5. To improve the quality of candidate solutions obtained from these mutational heuristics, Davis' Bit Hill Climbing is applied.

3.2 Experimental Settings

The experiments are performed on Pentium IV, 2 GHz Linux machines with 256 Mb memory. Fifty runs are performed for each hyper-heuristic and problem instance pair. For each problem instance, a set of fifty random initial configurations are created. Each run in an experiment is performed starting from the same initial configuration. The experiments are allowed to run for 600 CPU seconds. If the global optimum of the objective function is found before the time limit is exhausted, then the experiment is terminated.

The candidate solutions are encoded as bit strings. The continuous functions in benchmark set are encoded in Gray Code. The discrete functions have their own direct encoding. Foxhole Function has default dimension of 2. The default number of bits per dimension parameter is set to 8, 3, and 4 for the Royal Road, Goldberg, and Whitley Functions respectively. The rest of the functions have 10 dimensions and 30 bits are used to encode the range of a variable.

3.3 Experimental Results

The experimental results of performance comparison of 35 heuristic selection – acceptance criteria combinations on 14 different benchmark functions are statistically evaluated. For each benchmark function the combinations are sorted according to their performance. The average number of fitness evaluations needed to converge to global optimum is used as the performance criterion for the experiments with 100% success rate. The average best fitness reached is used for the experiments with success rates lower than 100%. The performances are evaluated statistically using t-test. Each combination has been given a ranking. Confidence interval is set to 95% in t-test to determine significant performance variance. The combinations that do not have significant performance variances are grouped together and have been given the same ranking. The average rankings of heuristic selection methods and move acceptance criteria are calculated to reflect their performance. In Table 2, average rankings for the heuristic selection methods are provided on each problem. The averages are obtained by testing the selection methods on each acceptance criteria. In Table 3, average rankings of acceptance criteria are given where the averages are obtained by testing acceptance criteria on each selection method this time. Lower numbers in these tables denote a higher placement in the ranking and indicate better performance. The average ranking of each selection method on all of the functions is depicted in Fig. 1, and the average ranking of each acceptance criterion on all of the functions in Fig. 2.

No heuristic selection and acceptance criterion couple came out to be a winner on all of the benchmark functions. *Choice Function* performs well on *Sphere* and *Griewangk* functions. *Simple Random* performs well on *Sphere Function*. *Random Descent* and *Random Permutation Descent* perform well on *Rotated Hyperellipsoid Function*. *Greedy* performs well on *Rosenbrock Function*. The performance variances of heuristic selection methods on remaining functions were not as significant as these

cases. Choice Function performs slightly better than remaining selection methods on average. *IE* acceptance criterion performs well on *Rastrigin*, *Schwefel*, *Easom, Rotated Hyperellipsoid*, and discrete deceptive functions. *OI* acceptance criterion performs well on *Rosenbrock Function*. *MC* acceptance criterion performs well on *Foxhole Function*. *IE* acceptance criterion indicates significantly a better performance than the remaining acceptance criteria on average.

Table 2. Average ranking of each selection method on each problem; *CF* stands for *Choice Function*, *SR* for *Simple Random*, *RD* for *Random Descent*, *RP* for *Random Permutation*, *RPD* for Random *Permutation Descent*, *Tabu* for *Tabu Search*, *GR* for *Greedy*.

Name	*CF*	*SR*	*RD*	*RP*	*RPD*	*TABU*	*GR*
Sphere	7.0	7.0	24.5	14.0	24.5	24.5	24.5
Rosenbrock	20.2	22.0	16.0	23.8	16.0	16.0	12.0
Step	17.7	17.7	17.7	18.9	17.7	17.7	18.6
Quartic *w/ noise*	17.9	17.9	17.9	17.9	17.9	17.9	18.6
Foxhole	15.7	15.7	15.7	19.3	15.7	15.7	28.2
Rastrigin	17.9	17.5	18.5	17.3	18.5	17.7	18.6
Schwefel	17.0	17.0	18.8	17.0	18.8	18.8	18.6
Griewangk	11.8	17.2	17.2	17.2	17.2	17.2	28.2
Ackley	16.5	16.5	16.5	23.5	16.5	16.5	20.0
Easom	16.0	16.0	21.7	16.0	21.7	21.7	12.9
Rotated Hyperellipsoid	20.4	21.2	13.4	21.6	14.8	19.8	15.6
Royal Road	16.8	17.6	17.1	17.4	17.1	17.8	22.2
Goldberg	18.6	19.3	16.6	19.4	17.4	16.1	18.6
Whitley	17.9	17.9	17.9	17.9	17.9	17.9	18.6

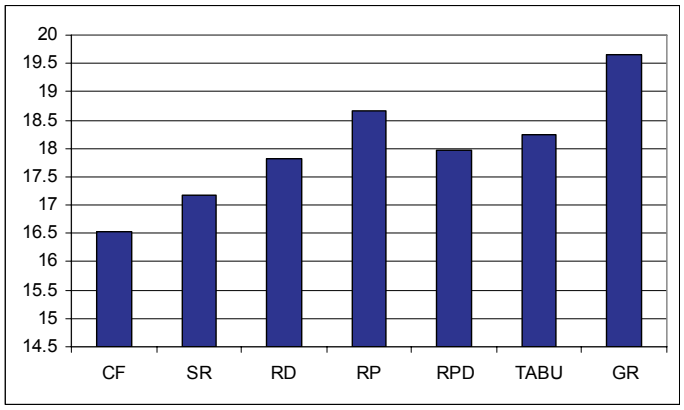

Fig. 1. Average ranking of each selection method on all problem instances

In Fig. 3 average number of evaluations to converge to global optimum by a selected subset of hyper-heuristics is depicted on a subset of benchmark functions, which are *Sphere, Ackley and Goldberg's Functions*. Fig. 3 (a), (c), and (e) visualize the performance comparison of the heuristic selection methods using IE acceptance criterion

for *Sphere, Ackley and Goldberg's Functions* respectively and Fig. 3 (b), (d), and (f) the performance comparison of the acceptance criteria using Choice Function heuristic selection method for *Sphere, Ackley and Goldberg's Functions* respectively. Lower average number of evaluations intends faster convergence to the global optimum and indicates better performance.

Table 3. Average ranking of each acceptance criterion on each problem; *AM* stands for *All Moves Accepted*, *OI* for *Only Improving Moves Accepted*, *IE* for *Improving and Equal Moves Accepted*, *MC* for *Monte Carlo Acceptance Criterion*, *and GD* for *Great Deluge Acceptance Criterion*.

Name	*AM*	*OI*	*IE*	*MC*	*GD*
Sphere	19.5	17.0	17.0	17.0	19.5
Rosenbrock	23.8	12.0	16.0	23.8	16.0
Step	29.1	18.6	17.7	18.9	17.7
Quartic *w/ noise*	29.1	17.4	14.5	14.5	14.5
Foxhole	12.4	27.7	26.5	11.1	12.4
Rastrigin	29.1	10.6	7.6	23.9	18.8
Schwefel	29.1	10.6	7.6	22.6	20.1
Griewangk	11.9	27.7	26.5	11.9	11.9
Ackley	19.0	19.0	16.5	16.5	19.0
Easom	23.3	11.6	8.5	23.3	23.3
Rotated Hyperellipsoid	25.1	11.7	8.8	22.4	22.6
Royal Road	28.1	10.6	7.6	23.0	20.7
Goldberg	29.1	10.6	7.6	22.4	20.4
Whitley	23.9	10.6	7.6	23.9	23.9

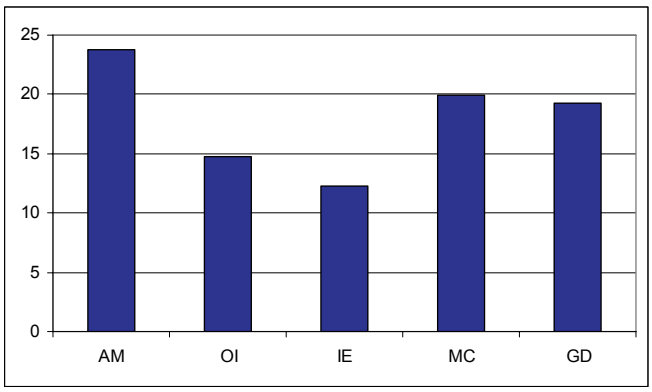

Fig. 2. Average ranking of each acceptance criterion on all problem instances

For *Sphere Model*, distinct performance variances are observed between heuristic selection methods in Fig. 3 (a) on the other side the difference is not so prominent between acceptance criteria in Fig. 3 (b). Fig. 3 (a) shows that Random Permutation and Choice Function heuristic selection methods achieved faster convergence than remaining selection methods. In Fig. 3 (c) and (d) it can be observed that Choice Function heuristic selection method and IE acceptance criterion accomplished a faster

convergence to global optimum on *Ackley Function*. Fig. 3 (e) and (f) show that Choice Function heuristic selection method and IE acceptance criterion performed best on *Goldberg's Function*. Fig. 3 (f) shows that the performance variances between different acceptance criteria are enormous on the same function. Also AM acceptance criterion cannot reach the global optimum on *Goldberg's Function* and no average number of evaluations to converge to global optimum value is depicted for this criterion in the same figure.

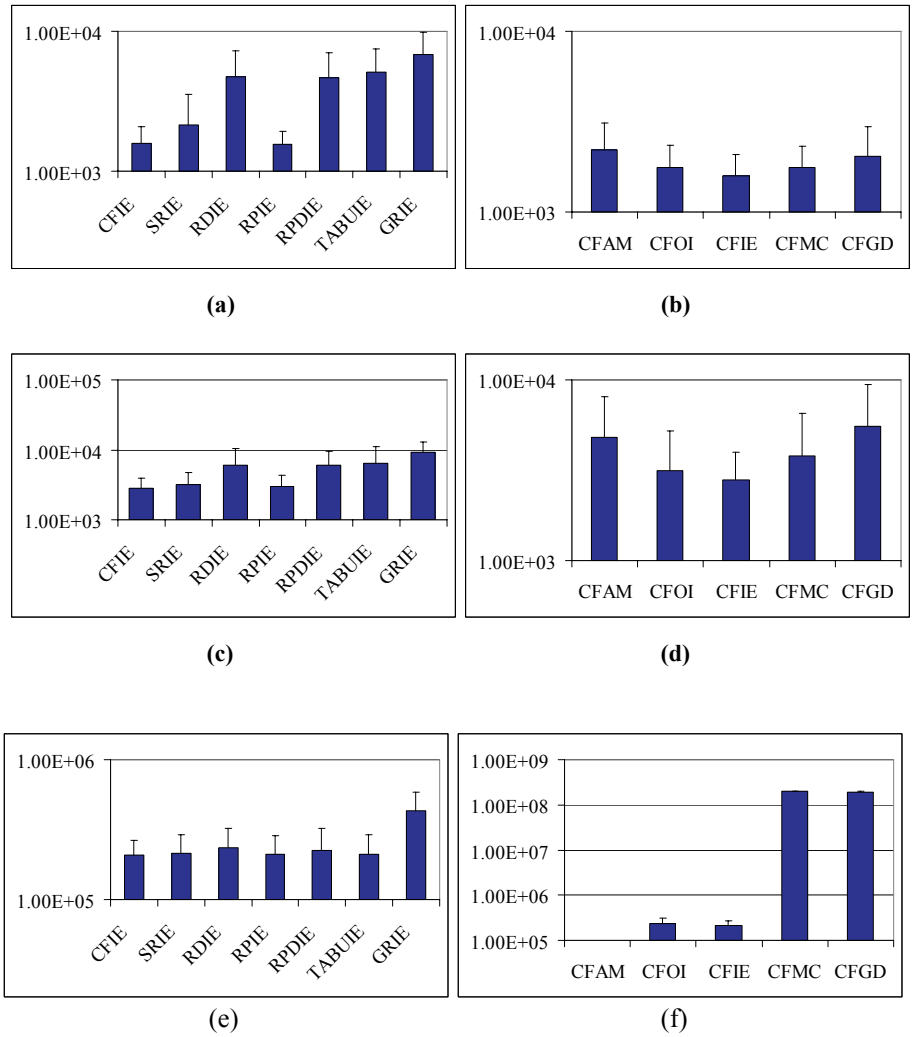

Fig. 3. Average number of evaluations to converge to global optimum of hyper-heuristics consisting of all heuristic selection methods using IE acceptance criterion on (a) *Sphere Model* function, (c) *Ackley Function* (e) *Goldberg Function*, and average number of evaluations to converge to global optimum of hyper-heuristics consisting of Choice Function heuristic selection method and all acceptance criteria on (b) *Sphere Model* function, (d) *Ackley Function* (f) *Goldberg Function*.

4 Hyper-heuristics for Solving Exam Timetabling Problems

4.1 Exam Timetabling Problem Instances and Settings

Carter's Benchmark [10] and Yeditepe University Faculty of Architecture and Engineering [25] data sets are used for the performance comparison of hyper-heuristics. The characteristics of as illustrated in Table 4.

Table 4. Parameters and properties of the exam timetabling problem instances

Instance	*Exams*	*Students*	*Enrollment*	*Density*	*Days*	*Capacity*
Carf92	543	18419	54062	0.14	12	2000
Cars91	682	16925	59022	0.13	17	1550
Earf83	181	941	6029	0.27	8	350
Hecs92	81	2823	10634	0.20	6	650
Kfus93	486	5349	25118	0.06	7	1955
Lsef91	381	2726	10919	0.06	6	635
Purs93	2419	30032	120690	0.03	10	5000
Ryes93	486	11483	45051	0.07	8	2055
Staf83	139	611	5539	0.14	4	3024
Tres92	261	4360	14901	0.18	10	655
Utas92	622	21267	58981	0.13	12	2800
Utes92	184	2749	11796	0.08	3	1240
Yorf83	190	1125	8108	0.29	7	300
Yue20011	140	559	3488	0.14	6	450
Yue20012	158	591	3706	0.14	6	450
Yue20013	30	234	447	0.19	2	150
Yue20021	168	826	5757	0.16	7	550
Yue20022	187	896	5860	0.16	7	550
Yue20023	40	420	790	0.19	2	150
Yue20031	177	1125	6716	0.15	6	550
Yue20032	210	1185	6837	0.14	6	550

Hyper-heuristics consisting of *Simple Random, Random Descent, Tabu Search, Choice Function, and Greedy* heuristic selection mechanisms and all the acceptance criteria, described in Section 2.1 are tested with each benchmark exam timetabling problem instance. The fitness function used for solving the exam timetabling problem takes a weighted average of the number of constraint violations. The fitness function is multiplied by -1 to make the problem a minimizing problem.

$$F(T) = \frac{-1}{1 + \sum_{\forall i} w_i g_i(T)} \quad (1)$$

In the equation **(1)**, w_i indicates the weight associated to the i^{th} constraint, g_i indicates the number of violations of i^{th} constraint for a given schedule T. The value 0.4 is used as the weight for the first and the third constraint and 0.2 for the second constraint as explained in Section 2.3.

4.1 Heuristics for Exam Timetabling

Candidate solutions are encoded as an array of timeslots where each locus represents an exam to be scheduled. Four heuristics are implemented to be used with the hyper-heuristics for solving an exam timetabling problem. Three of these heuristics utilize tournament strategy for choosing a timeslot to reschedule a given exam to improve a candidate solution based on a constraint type, while the last one is a mutation operator. Heuristics for the constraints (i) and (ii) work similarly. Each improving heuristic targets a different conflict. Both heuristics randomly choose a predetermined number of exams and select the exam with the highest number of targeted conflict among these. Also a predetermined number of timeslots are randomly chosen and the number of targeted conflicts are checked when the exam is assigned to that timeslot. The timeslot with the minimum number of targeted conflict is then assigned to the selected exam.

The heuristic which targets the capacity conflicts (iii) randomly chooses a predetermined number of timeslots and selects the timeslot with the maximum capacity conflict among these. A predetermined number of exams that are scheduled to this timeslot are chosen randomly and the exam that has the most attendants is selected among them. Again a group of timeslots are chosen randomly and the timeslot with the minimum number of attendants is assigned to the selected exam. Mutational heuristic passes over each exam in the array and assigns a random timeslot to the exam with a predetermined probability (*1/number of courses*).

4.2 Experimental Results

The experimental results of performance comparison of Simple Random, Random Descent, Tabu Search, Choice Function, and Greedy heuristic selection method and all acceptance criteria combinations on 21 different exam timetabling problem instances are statistically evaluated. Each pair has been assigned to a ranking. Confidence interval is set to 95% in t-test to determine the significant performance variance. Similar to the previous experiments, the combinations that do not have significant performance variances are assigned to the same ranking.

Average best fitness values for best performing heuristic selection-acceptance criterion combination are provided in Table 5. If several hyper-heuristics share the same ranking, than only one of them appears in the table, marked with *. Seven combinations that have the top average rankings are presented in Fig. 4. According to the results, Choice Function heuristic selection combined with Monte Carlo acceptance criterion has the best average performance on exam timetabling problems. The hyper-heuristic combinations with acceptance criteria AM and OI do not perform well on any of the problem instances.

Table 5. Average best fitness values for best performing heuristic selection-acceptance criterion combinations on each problem instance; *AM* stands for *All Moves Accepted*, *OI* for *Only Improving Moves Accepted*, *IE* for *Improving and Equal Moves Accepted*, *MC* for *Monte Carlo Acceptance Criterion*, *GD* for *Great Deluge Acceptance Criterion*.

Instance	(*Av. B. Fit., Std. Dev.*)	*H.Heuristic Alg.*
Carf92	(-1.02E-02, 1.18E-03)	TABU_IE *
Cars91	(-1.93E-01, 1.20E-01)	TABU_IE *
Earf83	(-7.27E-03, 4.94E-04)	CF_MC
Hecs92	(-2.19E-02, 2.43E-03)	CF_MC *
Kfus93	(-3.40E-02, 4.30E-03)	SR_GD
Lsef91	(-1.42E-02, 1.38E-03)	CF_MC
Purs93	(-1.41E-03, 6.98E-05)	SR_IE
Ryes93	(-1.08E-02, 1.37E-03)	CF_MC
Staf83	(-2.68E-03, 1.04E-05)	SR_MC *
Tres92	(-6.79E-02, 1.08E-02)	SR_GD
Utas92	(-1.87E-02, 1.79E-03)	TABU_IE *
Utes92	(-2.27E-03, 8.64E-05)	CF_MC
Yorf83	(-8.32E-03, 4.57E-04)	CF_MC
Yue20011	(-9.02E-02, 1.07E-02)	SR_GD
Yue20012	(-7.54E-02, 9.38E-03)	SR_GD
Yue20013	(-2.50E-01, 0.00E+00)	SR_MC *
Yue20021	(-3.45E-02, 4.55E-03)	SR_GD
Yue20022	(-1.26E-02, 9.08E-04)	CF_MC
Yue20023	(-1.52E-02, 2.69E-04)	CF_MC *
Yue20031	(-1.59E-02, 1.65E-03)	CF_MC
Yue20032	(-5.42E-03, 3.68E-04)	CF_MC

Table 6. The performance rankings of each heuristic selection-acceptance criterion on all problem instances. Lower rankings indicate better performance.

(a)

H.-h.	*Carf92*	*Cars91*	*Earf83*	*Hecs92*	*Kfus93*	*Lsef91*	*Purs93*
SR_AM	30.5	26.5	26	26	26	26	26
SR_OI	19.5	19	12.5	16	19	16	8
SR_IE	7.5	7.5	12.5	16	9	11.5	1
SR_MC	15	15	7	7.5	15	11.5	23
SR_GD	7.5	6	8	7.5	1	4.5	9
RD_AM	30.5	31.5	30	31	31	29.5	31.5
RD_OI	19.5	19	20	16	19	20	12.5
RD_IE	7.5	3	12.5	16	9	11.5	4
RD_MC	7.5	11.5	3.5	4.5	9	4.5	20.5
RD_GD	30.5	31.5	30	31	31	29.5	31.5
RP_AM	30.5	31.5	34.5	31	31	34.5	34.5
RP_OI	19.5	19	20	16	19	20	12.5
RP_IE	7.5	3	12.5	16	9	11.5	4
RP_MC	7.5	11.5	3.5	4.5	9	4.5	20.5
RP_GD	30.5	31.5	34.5	31	31	34.5	34.5

RPD_AM	30.5	31.5	30	31	31	29.5	31.5
RPD_OI	19.5	19	20	16	19	20	12.5
RPD_IE	7.5	3	12.5	16	9	11.5	4
RPD_MC	7.5	11.5	3.5	4.5	9	4.5	20.5
RPD_GD	30.5	31.5	30	31	31	29.5	31.5
CF_AM	30.5	26.5	30	31	31	33.5	27
CF_OI	19.5	19	20	16	19	20	12.5
CF_IE	7.5	3	12.5	16	9	11.5	4
CF_MC	7.5	9	1	1.5	3	1	16.5
CF_GD	19.5	19	20	16	19	20	12.5
TABU_AM	30.5	31.5	30	31	31	29.5	28.5
TABU_OI	19.5	19	20	16	19	20	12.5
TABU_IE	7.5	3	12.5	16	9	11.5	4
TABU_MC	7.5	11.5	3.5	4.5	9	4.5	20.5
TABU_GD	30.5	31.5	30	31	31	29.5	28.5
GR_AM	24.5	24.5	24	24.5	24.5	24.5	24.5
GR_OI	19.5	23	20	16	23	20	16.5
GR_IE	7.5	7.5	12.5	16	9	11.5	7
GR_MC	7.5	14	6	1.5	2	4.5	18
GR_GD	24.5	24.5	25	24.5	24.5	24.5	24.5

(b)

H.-h.	*Ryes93*	*Staf83*	*Tres92*	*Utas92*	*Utes92*	*Yorf83*
SR_AM	26	31	26	26	26	26
SR_OI	19.5	16	19.5	15	16	19.5
SR_IE	8	16	8.5	3.5	16	12
SR_MC	15	4.5	15	19	7	7
SR_GD	8	4.5	1	9	8	8
RD_AM	31	31	31	32.5	31	29.5
RD_OI	19.5	16	19.5	19	16	19.5
RD_IE	8	16	8.5	3.5	16	12
RD_MC	8	4.5	8.5	11.5	4	3.5
RD_GD	31	31	31	32.5	31	29.5
RP_AM	31	31	31	32.5	31	34.5
RP_OI	19.5	16	19.5	19	16	19.5
RP_IE	8	16	8.5	3.5	16	12
RP_MC	8	4.5	8.5	11.5	4	3.5
RP_GD	31	31	31	32.5	31	34.5
RPD_AM	31	31	31	32.5	31	29.5
RPD_OI	19.5	16	19.5	19	16	19.5
RPD_IE	8	16	8.5	3.5	16	12
RPD_MC	8	4.5	8.5	11.5	4	3.5
RPD_GD	31	31	31	32.5	31	29.5
CF_AM	31	26	31	27	31	33
CF_OI	19.5	16	19.5	19	16	19.5
CF_IE	8	16	8.5	3.5	16	12
CF_MC	1	4.5	2	8	1	1
CF_GD	19.5	16	19.5	19	16	19.5
TABU_AM	31	31	31	28.5	31	29.5
TABU_OI	19.5	16	19.5	19	16	19.5
TABU_IE	8	16	8.5	3.5	16	12
TABU_MC	8	4.5	8.5	11.5	4	3.5

TABU_GD	31	31	31	28.5	31	29.5
GR_AM	24.5	24.5	24.5	24.5	24.5	24.5
GR_OI	19.5	16	19.5	23	16	19.5
GR_IE	8	16	8.5	7	16	12
GR_MC	8	4.5	8.5	14	4	6
GR_GD	24.5	24.5	24.5	24.5	24.5	24.5

(c)

H.-h.	*Y011*	*Y012*	*Y013*	*Y021*	*Y022*	*Y023*	*Y031*	*Y032*
SR_AM	26	26	22.5	26	26	9.5	26	28.5
SR_OI	19.5	19.5	31.5	19.5	16	17.5	16	17.5
SR_IE	12	11.5	14	12	12	17.5	16	9
SR_MC	6	11.5	4	8	7.5	3.5	7.5	6.5
SR_GD	1	1	8	1	7.5	7	7.5	8
RD_AM	31	31	22.5	03	29.5	9.5	30	28.5
RD_OI	19.5	19.5	31.5	19.5	20	17.5	16	17.5
RD_IE	12	11.5	14	12	12	17.5	16	17.5
RD_MC	6	5	4	4.5	4	1.5	4	3.5
RD_GD	31	31	22.5	30	29.5	9.5	30	28.5
RP_AM	31	31	22.5	34.5	34.5	34.5	34.5	34.5
RP_OI	19.5	19.5	31.5	19.5	20	28	16	17.5
RP_IE	12	11.5	14	12	12	17.5	16	17.5
RP_MC	6	5	4	4.5	4	25	4	3.5
RP_GD	31	31	22.5	34.5	34.5	34.5	34.5	34.5
RPD_AM	31	31	22.5	30	29.5	31.5	30	28.5
RPD_OI	19.5	19.5	31.5	19.5	20	28	16	17.5
RPD_IE	12	11.5	14	12	12	17.5	16	17.5
RPD_MC	6	5	4	4.5	4	25	4	3.5
RPD_GD	31	31	22.5	30	29.5	31.5	30	32.5
CF_AM	31	31	22.5	30	33	9.5	30	32.5
CF_OI	19.5	19.5	31.5	19.5	20	17.5	16	17.5
CF_IE	12	11.5	14	12	12	17.5	16	17.5
CF_MC	3	5	4	4.5	1	1.5	1	1
CF_GD	19.5	19.5	31.5	19.5	20	17.5	16	17.5
TABU_AM	31	31	22.5	30	29.5	31.5	30	28.5
TABU_OI	19.5	19.5	31.5	19.5	20	28	16	17.5
TABU_IE	12	11.5	14	12	12	17.5	16	17.5
TABU_MC	6	5	4	4.5	4	25	4	3.5
TABU_GD	31	31	22.5	30	29.5	31.5	30	28.5
GR_AM	24.5	24.5	9.5	24.5	24.5	5.5	24.5	17.5
GR_OI	19.5	19.5	31.5	19.5	20	17.5	16	17.5
GR_IE	12	11.5	14	12	12	17.5	16	17.5
GR_MC	2	2	4	4.5	4	3.5	4	6.5
GR_GD	24.5	24.5	9.5	24.5	24.5	5.5	24.5	17.5

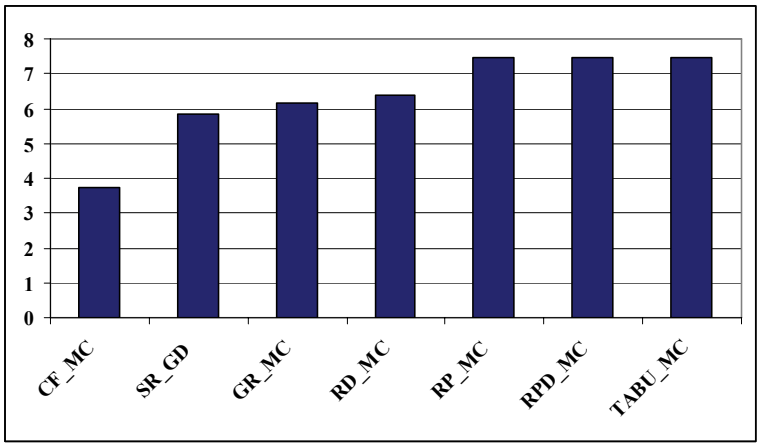

Fig. 4. Top seven heuristic selection method-acceptance criterion combinations considering the average ranking over all problem instances.

5 Conclusion

An empirical study on hyper-heuristics is provided in this paper. As an iterative search strategy, a hyper-heuristic is combined with a move acceptance strategy. Different such pairs are experimented on a set of benchmark functions. According to the outcome, experiments are expanded to cover a set of exam timetabling benchmark problem instances.

The experimental results denote that no combination of heuristic selection and move acceptance strategy can dominate over the others on all of the benchmark functions used. Different combinations might perform better on different objective functions. Despite this fact, IE heuristic acceptance criterion yielded better average performance. Considering heuristic selection methods, *Choice Function* yielded a slightly better average performance, but the difference between performance of *Choice Functio*n and other heuristic selection methods were not as significant as it was between acceptance criteria. The experimental results on exam timetabling benchmark indicated that *Choice Function* heuristic selection method combined with *MC* acceptance criterion performs superior than the rest of the hyper-heuristic combinations.

Acknowledgement

This research is funded by TUBITAK (The Scientific and Technological Research Council of Turkey) under grant number 105E027.

References

1. Ackley, D.: An Empirical Study of Bit Vector Function Optimization. Genetic Algorithms and Simulated Annealing, (1987) 170-215
2. Ayob, M. and Kendall, G.: A Monte Carlo Hyper-Heuristic To Optimise Component Placement Sequencing For Multi Head Placement Machine. Proceedings of the International Conference on Intelligent Technologies, InTech'03, Chiang Mai, Thailand, Dec 17-19 (2003) 132-141
3. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S.: Hyper-heuristics: an Emerging Direction in Modern Search Technology. Handbook of Metaheuristics (eds Glover F. and Kochenberger G. A.) (2003) 457-474
4. Burke, E., Newall, J. P., and Weare, R.F.: A Memetic Algorithm for University Exam Timetabling. Lecture Notes in Computer Science 1153 (1996) 241-250
5. Burke, E.K., Kendall, G., and Soubeiga, E: A Tabu-Search Hyper-heuristic for Timetabling and Rostering. Journal of Heuristics Vol 9, No. 6 (2003) 451-470
6. Burke, E., Elliman, D., Ford, P., and Weare, B.: Examination Timetabling in British Universities- A Survey. Lecture Notes in Computer Science, Springer-Verlag, vol. 1153 (1996) 76–90
7. Burke, E.K. and Newall, J.P. : Solving Examination Timetabling Problems through Adaption of Heuristic Orderings: Models and Algorithms for Planning and Scheduling Problems. Annals of Operations Research, vol. 129 (2004) 107-134
8. Burke, E.K., Meisels, A., Petrovic, S. and Qu, R.: A Graph-Based Hyper Heuristic for Timetabling Problems. Accepted for publication in the European Journal of Operational Research (2005)
9. Burke E.K., Petrovic, S. and Qu, R.: Case Based Heuristic Selection for Timetabling Problems. Accepted for publication in the Journal of Scheduling, Vol.9 No2. (2006)
10. Carter, M. W, Laporte, G., and Lee, S.T.: Examination Timetabling: Algorithmic Strategies and Applications. Journal of the Operational Research Society, 47 (1996) 373-383
11. Cowling P., Kendall G., and Soubeiga E.: A Hyper-heuristic Approach to Scheduling a Sales Summit. Proceedings of In LNCS 2079, Practice and Theory of Automated Timetabling III : Third International Conference, PATAT 2000, Konstanz, Germany, selected papers (eds Burke E.K. and Erben W) (2000) 176-190
12. Davis, L.: Bit Climbing, Representational Bias, and Test Suite Design, Proceeding of the 4th International conference on Genetic Algorithms (1991) 18-23
13. De Jong, K.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
14. Di Gaspero, L. and Schaerf, A.: Tabu Search Techniques for Examination Timetabling. Lecture Notes In Computer Science, selected papers from the Third International Conference on Practice and Theory of Automated Timetabling (2000) 104 - 117.
15. Easom, E. E.: A Survey of Global Optimization Techniques. M. Eng. thesis, Univ. Louisville, Louisville, KY (1990)
16. Even, S., Itai, A., and Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems. SIAM J. Comput., 5(4):691-703 (1976)
17. Gaw, A., Rattadilok P., and Kwan R. S. K.: Distributed Choice Function Hyperheuristics for Timetabling and Scheduling. Proc. of the 5th International Conference on the Practice and Theory of Automated Timetabling (2004) 495-498
18. Goldberg, D. E.: Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. Complex Systems (1989) 129-152
19. Goldberg, D. E.: Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. Complex Systems (1989) 153-171

20. Griewangk, A.O.: Generalized Descent of Global Optimization. Journal of Optimization Theory and Applications, 34: 11.39 (1981)
21. Kendall G. and Mohamad M.: Channel Assignment in Cellular Communication Using a Great Deluge Hyper-heuristic, in the Proceedings of the 2004 IEEE International Conference on Network (ICON2004)
22. Marin, H. T.: Combinations of GAs and CSP Strategies for Solving Examination Timetabling Problems. Ph. D. Thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey (1998)
23. Merlot, L.T.G., Boland, N., Hughes, B. D., and Stuckey, P.J.: A Hybrid Algorithm for the Examination Timetabling Problem. Proc. of the 4th International Conference on the Practice and Theory of Automated Timetabling (2002) 348-371
24. Mitchell, M., and Forrest, S.: Fitness Landscapes: Royal Road Functions. Handbook of Evolutionary Computation, Baeck, T., Fogel, D., Michalewiz, Z., (Ed.), Institute of Physics Publishing and Oxford University (1997)
25. Ozcan, E., and Ersoy, E.: Final Exam Scheduler - FES, Proc. of 2005 IEEE Congress on Evolutionary Computation, vol. 2, (2005) 1356-1363
26. Ozcan, E., Towards an XML based standard for Timetabling Problems: TTML, Multidisciplinary Scheduling: Theory and Applications, Springer Verlag, (2005) 163 (24)
27. Paquete, L. F. and Fonseca, C. M.: A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms. Proc. of the 4th Metaheuristics International Conference (MIC 2001) 149-154
28. Petrovic, S., Yang, Y., and Dror, M.: Case-based Initialisation for Examination Timetabling. Proc. of 1st Multidisciplinary Intl. Conf. on Scheduling: Theory and Applications (MISTA 2003), Nottingham, UK, Aug 13-16 (2003) 137-154
29. Rastrigin, L. A.: Extremal Control Systems. In Theoretical Foundations of Engineering Cybernetics Series, Moscow, Nauka, Russian (1974)
30. Schwefel, H. P.: Numerical Optimization of Computer Models, John Wiley & Sons (1981), English translation of Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie (1977)
31. Whitley, D.: Fundamental Principles of Deception in Genetic Search. In G. J. E. Rawlins (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Matco, CA (1991)
32. Wong, T., Côté, P. and Gely, P.: Final Exam Timetabling: A Practical Approach. Proc. of IEEE Canadian Conference on Electrical and Computer Engineering, Winnipeg, CA, May 12-15, vol. 2 (2002) 726-731