Hill Climbers and Mutational Heuristics in Hyperheuristics

Ender Özcan, Burak Bilgin, Emin Erkan Korkmaz

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

Abstract. Hyperheuristics are single candidate solution based and simple to maintain mechanisms used in optimization. At each iteration, as a higher level of abstraction, a hyperheuristic chooses and applies one of the heuristics to a candidate solution. In this study, the performance contribution of hill climbing operators along with the mutational heuristics are analyzed in depth in four different hyperheuristic frameworks. Four different hill climbing operators and three mutational operators are used during the experiments. Various subsets of the heuristics are evaluated on fourteen well-known benchmark functions.

1 Introduction

The term hyperheuristics refers to a recent approach in search methodologies [2, 4, 5, 7, 17, 23]. The hyperheuristic concept involves a higher level of abstraction than metaheuristic methods. This term describes an iterative search approach which controls a set of heuristics. The method keeps track of the non problem-specific data such as the fitness change, the execution time and applies a heuristic at each iteration. Studies involving a number of heuristic selection and acceptance mechanism combinations are reported in the literature [2, 3, 4, 7, 17]. A comprehensive study on the performance of different heuristic selection and move acceptance strategies is reported in [3].

In this paper, the synergy of various heuristics and their contribution to the performance is evaluated on a set of benchmark functions. Furthermore, four different hyperheuristic frameworks that utilize a set of hill climbers as heuristics in addition to a set of mutational heuristics, are defined and assessed as well. The new frameworks are derived from the commonly used framework. The intention of this study is to answer the following questions: What type of heuristics is useful to be used in hyperheuristics? Do the hill climbers improve the performance if used within hyperheuristics? Can we use only hill climbers as heuristics? At which stage(s) and how can hill climbers be used to improve the performance? Is it possible to identify the problem domains where a specific framework might perform better as compared to the others?

2 Preliminaries

In general, exhaustive methods are impractical for solving real world problems, whereas meta-heuristics provide better means by intelligently seeking optimal solutions within a search space. For many practical problems meta-heuristics provide state-of-the-art solutions. Their success is due to the problem-specific implementations, which utilize knowledge about the problem domain and properties. The deployment of meta-heuristics requires expert level knowledge and experience on the problem tackled. Furthermore, fine tuning might be required [4, 23]. Hyperheuristics are general search methods that can be applied to any optimization problem easily [7]. Hyperheuristics describe a set of strategies that are used to choose a heuristic from a set of low level heuristics as illustrated in Fig. 1. There are very simple strategies that can be coded easily. Yet, a meta-heuristic can be used as a heuristic underneath a hyperheuristic as well as a hyperheuristic itself within this framework.

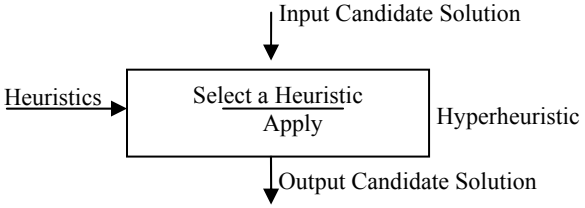


Fig. 1. Traditional hyperheuristic framework

Hyperheuristics operate on the search space of heuristics instead of candidate solutions. Non problem-specific data like heuristic execution time and changes in the fitness function can be used by hyperheuristics to select and apply a heuristic [2]. Although the methods of this type are reported in the literature before, the term hyperheuristic is first proposed by Cowling et al. [7] to name this approach. The early studies date back to Fisher and Thompson. They used a hyperheuristic based on probabilistic weighting of heuristics to solve the job-shop scheduling problem [12]. Kitano [19] used a genetic algorithm as a hyperheuristic for designing neural network topology. The hyperheuristic approach is utilized by Gratch et al. [15] to schedule earth-orbiting satellites and ground stations communications. Fang et al. [11] utilized this approach using the genetic algorithm to tackle the open-shop problem. Hart and Ross [17] tackled the dynamic job-shop problem with a similar approach. Hyperheuristics are applied to university exam timetabling problems by Terashima-Marin et al. [25].

A single iteration of a hyperheuristic method can be decomposed in two stages, heuristic selection and movement acceptance. In the previous studies, hyperheuristics might be named without discriminating between heuristic selection and acceptance criterion. Examples of heuristic selection methods are Simple, Greedy, Choice Function [7], Tabu-Search [5], and Case Based Heuristic Selection Methods [6]. Simple Hyperheuristics utilize randomized processes to select heuristics. Greedy Hyperheuristic chooses the best performing heuristic at each iteration. Choice Function Hyperheuristic keeps track of previous performance of each heuristic and makes a choice

between them by evaluating their performance via a choice function. Two types of deterministic acceptance criteria are used in [5, 7]: All Moves Accepted (AM) and Only Improving Moves Accepted (OI). Non-deterministic acceptance criteria can be found in [2, 17]. Monte Carlo Hyperheuristic accepts all of the improving moves and the non-improving moves can be accepted based on a probabilistic framework [2]. Great Deluge Hyperheuristic utilizes the Great Deluge Algorithm as the acceptance criterion [17]. Monte Carlo and Great Deluge Hyperheuristics both use Simple Random as heuristic selection method in [2, 17]. An experimental study on the performance of various heuristic selection and acceptance criterion combinations yielded that the combination of Choice Function, Improving and Equal Moves Accepted (IE) strategy and bit modifying heuristics performed the best on benchmark functions [3]. Hence, during the experiments this combination is used.

2.1 Benchmark Functions

A set of benchmark functions can be used to represent a broad range of optimization problems with various fitness landscapes. For the performance evaluation of different heuristic sets within different hyperheuristic frameworks, fourteen different benchmark functions are utilized. Characteristics of each benchmark function and the sources where they are obtained are summarized in Table 1.

Table 1. Characteristics of benchmark functions: *lb* indicates lower bound, *ub* upper bound, *opt* optimum point, *dim* number of dimensions, *bits* number of bits per dimension, *Conti.* continuity, *Cont.* continuous, *Disc.* discrete, and *Multi*. multimodal

Label	*lb*	*ub*	*opt*	*dim*	*bits*	*Conti.*	*Modality*	*Source*
F1	-5.12	5.12	0	10	30	Cont.	Unimodal	[8]
F2	-2.048	2.048	0	10	30	Cont.	Unimodal	[8]
F3	-5.12	5.12	0	10	30	Cont.	Unimodal	[8]
F4	-1.28	1.28	1	10	30	Cont.	Multi.	[8]
F5	-65.536	65.536	0	2	30	Cont.	Multi.	[8]
F6	-5.12	5.12	0	10	30	Cont.	Multi.	[15]
F7	-500	500	0	10	30	Cont.	Multi.	[16]
F8	-600	600	0	10	30	Cont.	Multi.	[12]
F9	-32.768	32.768	0	10	30	Cont.	Multi.	[1]
F10	-100	100	-1	10	30	Cont.	Unimodal	[1]
F11	-65.536	65.536	0	10	30	Cont.	Unimodal	[8]
F12	-	-	0	8	8	Disc.	-	[14]
F13	-	-	0	30	3	Disc.	-	[10]
F14	-	-	0	6	4	Disc.	-	[18]

2.2 Heuristics for Benchmark Function Optimization

Heuristics are classified as *mutational heuristics* and *hill climbers* in this paper. Hill climbers generate a *better* output candidate solution as a local search component, after they are applied to an input candidate solution. Mutational heuristics do not necessarily generate a better output candidate solution. 4 hill climbing algorithms and

3 mutational heuristics are implemented as heuristics to be used for solving binary encoded problems.

Hill climbing algorithms are as follows: Davis' Bit Hill Climbing Algorithm (DBHC) [8], Next Descent Hill Climbing Algorithm (NDHC) Random Bit Hill Climbing Algorithm (RBHC) and Steepest Descent Hill Climbing Algorithm (SDHC) [19]. All hill climbers make a set of modifications on a given candidate solution and each modification is accepted if there is an improvement in the generated solution. Assuming that a candidate solution is represented by a binary string, in each NDHC step a bit is inverted. The whole string is scanned bit by bit starting from the first until to the last. A DBHC differs from NDHC due to the scanning order. DBHC predetermines a random sequence to apply a hill climbing step and scans through the candidate solution according to it. During each RBHC step a bit is selected randomly and inverted for a number of iterations. SDHC checks each single bit inversion variant of the input candidate and accepts the one with the *best* improvement.

Mutational heuristics are Swap Dimension (SWPD), Dimensional Mutation (DIMM) and Hyper-mutation (HYPM). Swap Dimension heuristic randomly chooses two different dimensions in a candidate solution and swaps them. Dimensional Mutation heuristic randomly chooses a dimension and inverts all bits in this dimension with a probability of 0.5. Hyper-mutation randomly inverts each bit in the candidate solution with a probability of 0.5.

2.3 Hyperheuristic Frameworks

Recent studies presented in [21] shows that in memetic algorithms, using a single efficient hill climber instead of using a set of hill climbers where the operator selection is carried out self adaptively, might yield better solutions. As a result, different frameworks based on the general hyperheuristic approach can be defined in order to make better use of hill climbers as heuristics. In this study, four different frameworks are used; F_A, F_B, F_C and F_D, as summarized in Fig. 2.

F_A is the traditional framework and the others are the newly proposed ones. Hill climbers are used together with the mutational hill climbers. In some situations, after applying a mutational heuristic a hill climbing might be desirable. For example, if IE is used in the hyperheuristic, then most of the mutational heuristic moves will be declined. To avoid this phenomenon and to make better use of diversity provided by mutational heuristics, a hill climber can be utilized additionally. F_B represents such a framework. If the hyperheuristic chooses a mutational heuristic, then a predefined single hill climber is applied to the candidate solution. Notice that F_B still uses all heuristics together. In F_C, hill climbers are separated from the mutational heuristics. Hyperheuristic chooses only an appropriate mutational heuristic. Application of a selected heuristic to a candidate solution is followed by a hill climbing. A single hill climber is predefined by the user. F_D is a more general form of F_C. Two hyperheuristic modules are used; one for selecting an appropriate mutational heuristic and one for selecting an appropriate hill climber. F_D can be implemented in two ways. The acceptance mechanism of the hyperheuristic for hill climbers can get a feedback from the

intermediate candidate solution (Fig. 2- F_D, marked solid lines) or from the initial candidate solution (Fig. 2-F_D, dashed line).

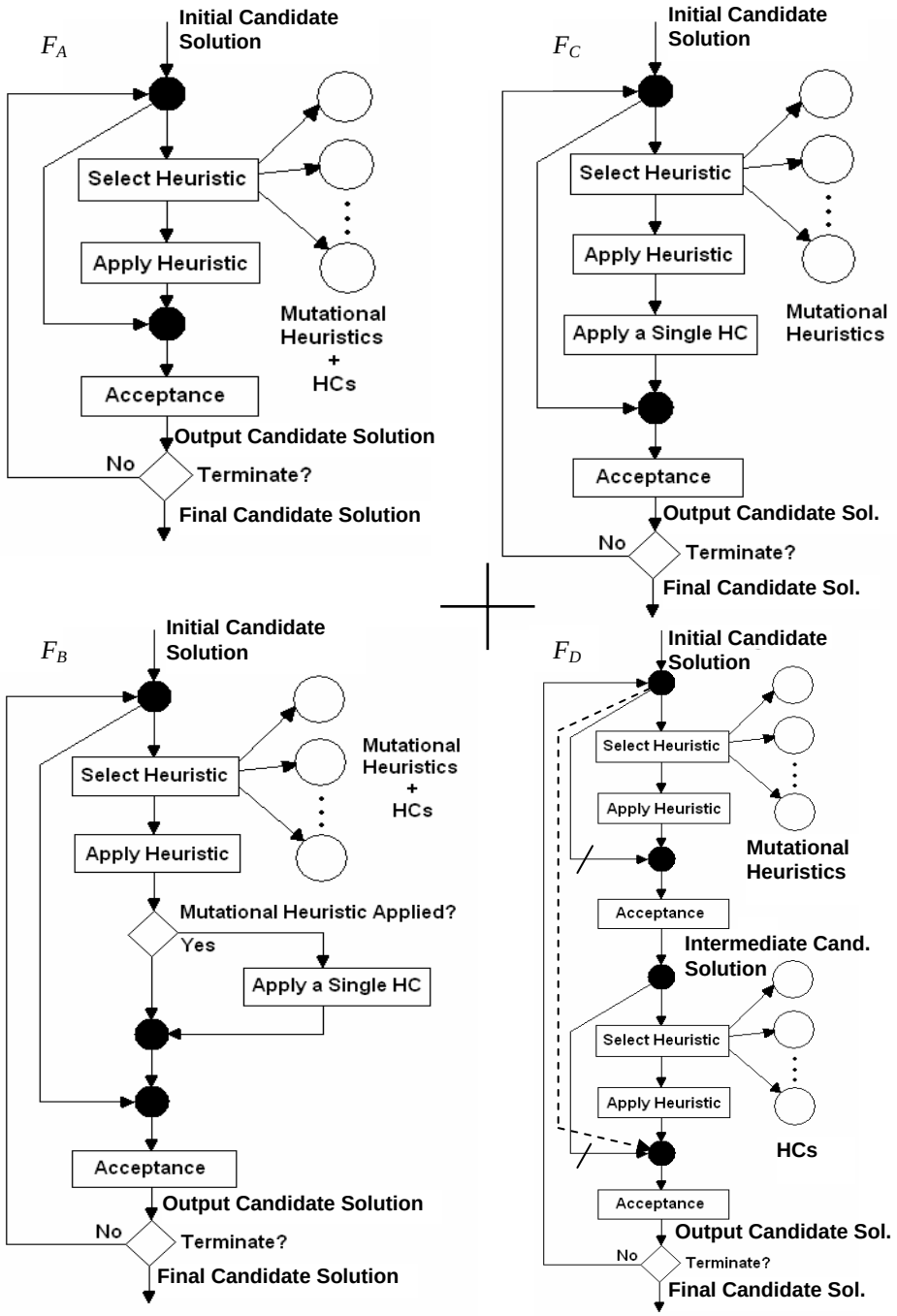


Fig. 2. Different hyperheuristic frameworks combining mutational heuristics and hill climbers.

3 Experiments

The experiments are performed on Pentium IV, 2 GHz Linux machines with 256 Mb memory. Fifty runs are performed for each heuristic set and problem instance pair. For each problem instance, a set of fifty random initial candidate solutions are created. Each run in an experiment is performed starting from the same initial candidate solution. 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 than the experiment is terminated.

3.1 Experimental Settings

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. Linear combinations of deceptive function variables are created to make them multidimensional. F5 has default dimension of 2. The default number of bits per dimension parameter is set to 8, 3, and 4 for the F12, F13, and F14 respectively. The rest of them have 10 dimensions and 30 bits are used to encode a variable (Table 1).

Hyperheuristic pattern is defined as the set of heuristics and the framework utilized in a hyperheuristic algorithm. The experimental set consists of eleven different hyperheuristic patterns; H1-H11 (Table 2). The frameworks F_A and F_B are tested combining hill climbers (HCs) with each mutational heuristic to observe the contribution of each one. If just hill climbers are used without having any mutational heuristics in the system, then both frameworks F_B and F_D reduce to F_A and F_C becomes local search. Hyperheuristic patterns are tested on 14 different benchmark functions. Choice Function and IE pair is used as a hyperheuristic during the experiments, except for H11. This pair is used on hill climbers, while Simple Random and AM pair is used on mutational heuristics. The single hill climber within the frameworks F_B and F_C is chosen as *DBHC* during the experiments.

3.2 Experimental Results

The runs, where the global optimum is found before time limit is exceeded, are considered to be successful. *Success rate*, the ratio of successful runs to all runs, is used as a performance criterion. There exists at least one hyperheuristic pattern that obtains an optimal solution during the runs for each benchmark function, except F4, which represents a search space with noise.

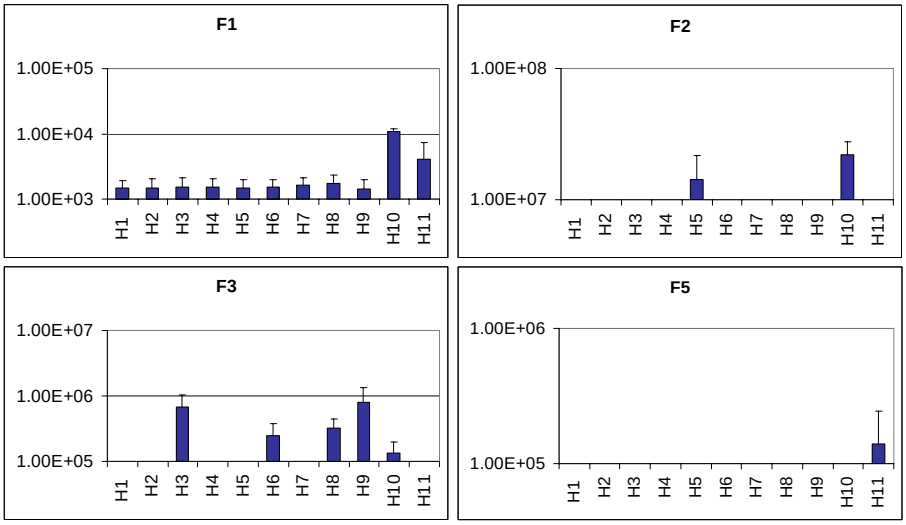
The average number of evaluations of the hyperheuristic patterns achieving full success during all runs on each benchmark function is depicted in Fig. 3. The traditional framework F_A with hill climbers performed poorly on most of the benchmark functions. There is always a better framework than F_A for all cases, except for F1. Even for F1, the performance of F_A is not significantly better than the rest. The framework F_B with all hill climbers and *SWPD* heuristic performed well on the benchmark functions F2, F9, and F11. These functions carry epistasis between di-

mensions. The experiments with F_A, F_B and each mutational heuristics showed that in some cases a good choice of mutational heuristics might yield a better performance. For example, *DIMM* provided a significantly better performance compared to the rest of the mutational heuristics in F_A and F_B for F6 and F11. Furthermore, in some cases, the framework might generate a synergy between operators providing an improved performance. For example, F_B performed significantly better than F_A when all hill climbers are used and *SWPD* in F2 and F8.

Table 2. Heuristic set and the framework used in each hyperheuristic pattern; H1-H11, where + and * indicate that the corresponding heuristic is controlled by the same hyperheuristic and – points out the heuristic that is used as the single hill climber within the related framework

Sets:	H1	H2	H3	H4	H5	H6	H7	**H8**	**H9**	**H10**	**H11**
NDHC	+	+	+	+	+	+	+	+	+		+
DBHC	+	+	+	+	+/–	+/–	+/–	+/–	+	–	+
RBHC	+	+	+	+	+	+	+	+	+		+
SDHC	+	+	+	+	+	+	+	+	+		+
SWPD		+			+			+	+	+	*
DIMM			+			+		+	+	+	*
HYPM				+			+	+	+	+	*
Framework	F_A	F_A	F_A	F_A	F_B	F_B	F_B	F_B	F_A	F_C	F_D

The framework F_C with all mutational heuristics and *DBHC* hill climber performed well on F3, F6, F7, and F10 which are continuous benchmark functions either unimodal or multimodal. Furthermore, F_C yielded a significantly better performance in solving discrete deceptive problems as compared to the rest. F_D was the only framework generating full success in all the runs for F5. This framework also performed well on F8. Both functions represent continuous and highly multimodal search spaces.



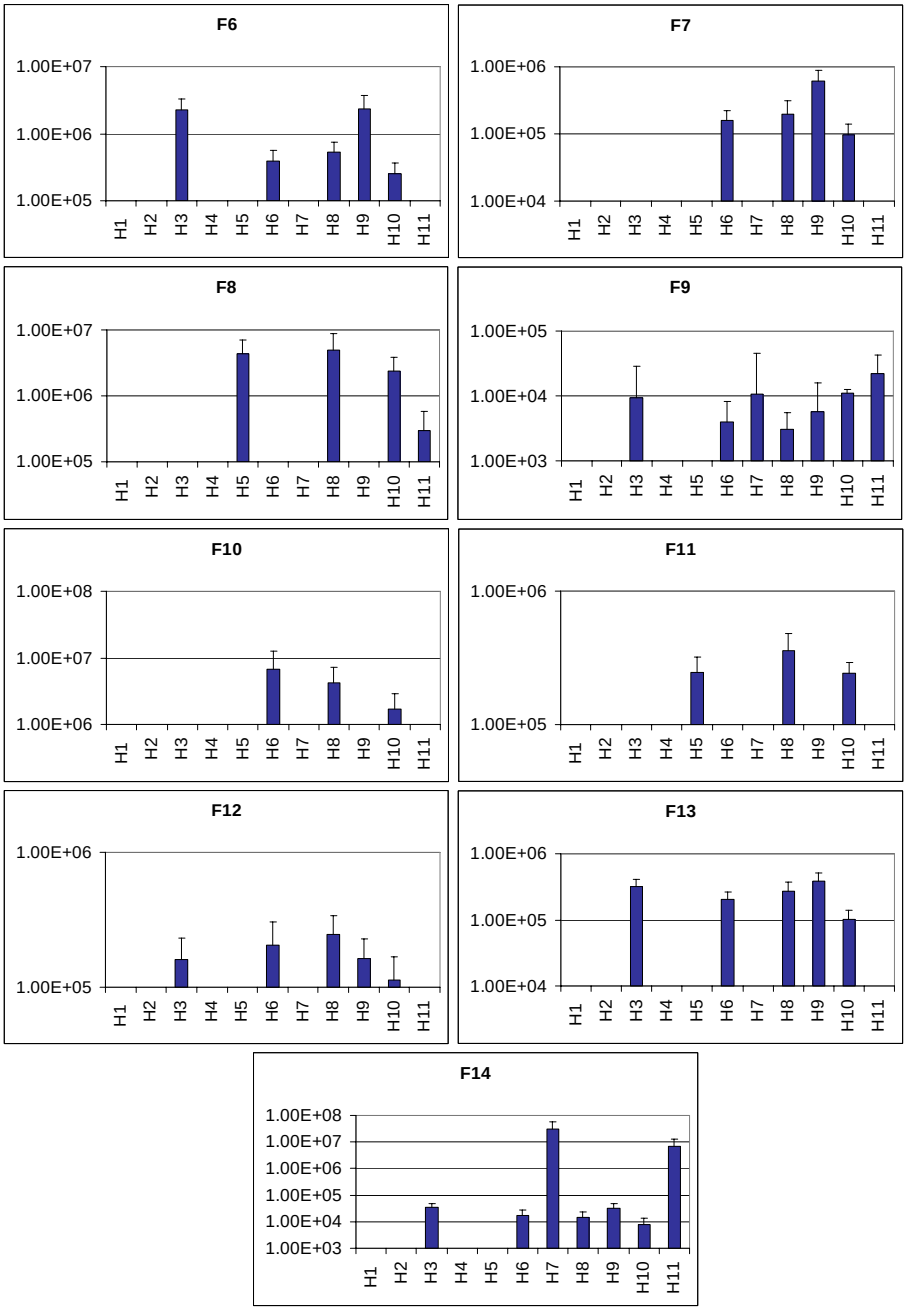


Fig. 3. Average number of fitness evaluations and their standard deviations for each hyperheuristic pattern having full success in all the runs.

4 Conclusion

The traditional hyperheuristics framework is extended to embed hill climbers as heuristics in various ways. Three different new frameworks are proposed and experimented on a set of benchmark functions, together with the traditional framework. Additionally, in order to observe the effects of mutational heuristics combined with hill climbers, an extra set of experiments is arranged.

The empirical results indicate that two of the newly proposed frameworks F_B and F_C have a better average performance than the traditional one. The third framework proposed turns out to be significantly successful for solving two highly multimodal benchmark functions compared to the rest. Furthermore, a hyperheuristic framework does not perform well, if it contains only hill climbers as heuristics. Obviously, most of the problems require utilization of a mutational heuristic in order not to get stuck at local optima. It has been observed that the choice of heuristics, whether it is a hill climber or a mutational heuristic determines the performance along with the choice of the framework. Exploitation and exploration capability of a hyperheuristic algorithm is determined by the heuristics used within. Mutational heuristics and hill climbers, combined underneath a decent framework might generate a synergy, yielding a better performance.

It seems that the traditional perturbation approaches are appropriate to be used as mutational heuristics. For example, random perturbation of a locus in a candidate solution, similar to mutation in evolutionary algorithms seems to perform well. Dimensional or content swapping operators can be helpful. Even, a hypermutation like heuristic, generating a random candidate solution might become handy, especially, whenever a hill climber is invoked afterwards. In our experiments, *SWPD* was useful in the benchmark problems with interdimensional epistasis, while *DIMM* and *HYPM* were very useful in multimodal benchmark functions to escape from the local optima.

Acknowledgement

This research is funded by TUBİTAK (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 Hyperheuristic To Optimise Component Placement Sequencing For Multi Head Placement Machine. Proc. of the Int. Conference on Intelligent Technologies, InTech'03, Chiang Mai, Thailand, Dec 17-19 (2003) 132-141
3. Bilgin, B., Ozcan, E., Korkmaz, E.E., An Experimental Study on Hyper-heuristics and Final Exam Scheduling, Proc. of the 6th Int. Conf. on PATAT 2006, to appear (2006)
4. Burke, E. K., Kendall, G., Newall, J., Hart E., Ross, P., Schulenburg, S.: Hyperheuristics: An Emerging Direction in Modern Search Technology. In: Glover, F., Kochenberger, G. A.

(eds.): Handbook of Metaheuristics. International Series in OR & Management Science, Vol. 57. Kluwer Academic Publishers, Boston Dordrecht London (2003) 457–474
5. Burke, E.K., Kendall, G., and Soubeiga, E: A Tabu-Search Hyperheuristic for Timetabling and Rostering. Journal of Heuristics Vol 9, No. 6 (2003) 451-470
6. Burke E.K., Petrovic, S. and Qu, R.: Case Based Heuristic Selection for Timetabling Problems. Journal of Scheduling, Vol.9 No2. (2006) 99-113
7. Cowling P., Kendall G., and Soubeiga E.: A Hyperheuristic Approach to Scheduling a Sales Summit. LNCS vol. 2079, PATAT 2000, selected papers (2000) 176-190
8. Davis, L.: Bit Climbing, Representational Bias, and Test Suite Design, Proceeding of the 4th International conference on Genetic Algorithms (1991) 18-23
9. De Jong, K.: An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan (1975)
10. Easom, E. E.: A Survey of Global Optimization Techniques. M. Eng. thesis, Univ. Louisville, Louisville, KY (1990)
11. Fang, H-L., Ross, P. M. and Corne, D.: A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems. Proc. of the 11th European Conf. on Artificial Intelligence (1994) 590-594
12. Fisher H. and Thompson, G. L.: Probabilistic Learning Combinations of Local Job-shop Scheduling Rules. Factory Scheduling Conf., Carnegie Institute of Tech., May 10-12 1961.
13. Goldberg, D. E.: Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. Complex Systems (1989) 129-152
14. Goldberg, D. E.: Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. Complex Systems (1989) 153-171
15. Gratch, J., Chein, S. and de Jong, G.: Learning Search Control Knowledge for Deep Space Network Scheduling. Proc. of 10th Int. Conf. on Machine Learning (1993) 135-142
16. Griewangk, A.O.: Generalized Descent of Global Optimization. Journal of Optimization Theory and Applications, 34 (1981) 11-39
17. Hart, E., and Ross, P. M.: A Heuristic Combination Method for Solving Job-Shop Scheduling Problems. PPSN V, LNCS Vol. 1498, Springer Berlin (1998) 845-854
18. Kendall, G. and Mohamad M.: Channel Assignment in Cellular Communication Using a Great Deluge Hyperheuristic, Proc. of the IEEE Int. Conf. on Network (2004) 769-773
19. Kitano, H.: Designing Neural Networks Using Genetic Algorithms with Graph Generation Systems. Complex Systems 4(4) (1990) 461-476
20. Mitchell, M., and Forrest, S.: Fitness Landscapes: Royal Road Functions. Handbook of Evolutionary Computation, Baeck, T., Fogel, D., Michalewiz, Z., (eds.), Institute of Physics Publishing and Oxford University (1997) 1-25
21. Ozcan, E.: An Empirical Investigation on Memes, Self-generation and Nurse Rostering, Proc. of the 6th Int. Conf. on PATAT 2006, to appear (2006)
22. Rastrigin, L. A.: Extremal Control Systems. In Theoretical Foundations of Engineering Cybernetics Series, Moscow, Nauka, Russian (1974)
23. Ross, P.: Hyperheuristics. In: Burke, E. K., Kendall, G. (eds.): Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer-Verlag, Berlin Heidelberg New York (2005) 529–556
24. Schwefel, H. P.: Numerical Optimization of Computer Models, John Wiley & Sons (1981), English translation of Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie (1977)
25. Terashima-Marin, H., Ross, P. M., and Valenzuela-Rendon, M.: Evolution of Constraint Satisfaction Strategies in Examination Timetabling. Proc. of Genetic and Evolutionary Computation Conference – GECCO, (1999) 635-642
26. Whitley, D.: Fundamental Principles of Deception in Genetic Search. In G. J. E. Rawlins (eds.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Matco, CA (1991)