LANDSCAPE ANALYSIS OF SIMPLE PERTURBATIVE HYPER-HEURISTICS

İbrahim Maden

Istanbul Technical University
Informatics Institute
34469 Maslak Istanbul
Turkey
imaden@itu.edu.tr

Şima Uyar

Istanbul Technical University
Computer Engineering Department
34469 Maslak Istanbul
Turkey
etaner@itu.edu.tr

Ender Özcan

University of Nottingham
ASAP Research Group
School of Computer Science
Jubilee Campus
NG8 1BB Nottingham
United Kingdom
exo@cs.nott.ac.uk

Abstract: *Hyper-heuristics introduce novel approaches for solving hard combinatorial optimization problems. A hyper-heuristic method operates over a set of low level heuristics. There are different hyper-heuristic frameworks that employ the idea of automating the heuristic design process. In a perturbative hyper-heuristic framework, the most appropriate low level heuristic is automatically determined and applied to solve a given problem at each step of the search process. A landscape analysis technique provides means for understanding the influence of operators and algorithmic behavior for a given problem. In this study, we aim to understand and analyze a set of perturbative hyper-heuristics through landscape analysis based on an auto-correlation function. Tests are performed on a series of commonly used benchmark functions. To the best of the authors' knowledge, no such prior landscape analysis exists in literature for the perturbative hyper-heuristics.*

Keywords: *hyper-heuristic, landscape analysis, correlation length, heuristic performance*

1 Introduction

Hyper-heuristics [8,19,22] are novel approaches for solving hard combinatorial optimization problems. A hyper-heuristic method operates over a set of low level heuristics. Hyper-heuristics separate the problem domain and the solution domain from each other operating on the low level heuristics. They do not require any problem specific information. There are different hyper-heuristic frameworks that employ the idea of automating the heuristic design process. In a perturbative hyper-heuristic framework, a hyper-heuristic repeatedly chooses and applies low level heuristics moving towards better regions in the search space until the end of iterations. The output of a hyper-heuristic is the best solution found during the search. As a result, hyper-heuristics can be thought of as black box search techniques that get the problem instance and a list of low level heuristics as inputs, run until the stopping criteria are satisfied and finally output the solution. In literature, hyper-heuristics can be categorized in two main groups [8], namely heuristics to choose heuristics and heuristics to generate heuristics, both of which can be further be divided into two more classes, i.e. perturbative hyper-heuristics and constructive hyper-heuristics. These different hyper-heuristic methods will be explained further in Section 2. In this study, we focus on hyper-heuristic approaches which use perturbative low-level heuristics and fall in the category of heuristics to choose heuristics.

Fitness denotes the objective function value which measures the quality of a solution to a given problem. A *fitness landscape* shows the association between the fitness space and the search space. By the help of fitness landscape analysis, performance of heuristics on a problem can be predicted and more effective operators can be designed. One of the most commonly used metrics to analyze landscapes is the “auto correlation function” or the “correlation length” which is derived from it. A high correlation length means a smooth landscape and a low correlation length shows a rugged landscape. Smooth landscapes are easier for search algorithms that rely on the neighborhood structure of the underlying landscape.

In this study, we focus on analyzing the landscape properties of perturbative hyper-heuristics. To the authors' best knowledge, no such prior landscape analysis exists in literature for this type of hyper-heuristics. We use the "fitness correlation length" metric to analyze the landscape of a simple perturbative hyper-heuristic framework. Tests are performed on seven well known benchmark functions.

The rest of the paper is organized as follows: Section 2 gives an overview on hyper-heuristics and fitness landscape analysis. In section 3, the proposed method and framework for analyzing the landscape of the

perturbative hyper-heuristics is explained. The experiments and the results are given in Section 4. Section 5 concludes the paper and provides directions for future work.

2 Hyper-heuristics and Landscape Analysis

2.1 Hyper-heuristics

Hyper-heuristics [8,19,22] operate on a search space of heuristics rather than directly on a search space of solutions. Cowling et al. [2] also define hyper-heuristics as "an approach that operates at a higher level of abstraction than metaheuristics and manages the choice of which low-level heuristic method should be applied at any given time, depending upon the characteristics of the region of the solution space currently under exploration". In other words, a hyper-heuristic itself does not search for a better solution for the problem. Instead, it selects at each step of the solution process, the most promising simple low-level heuristic which has a high potential to improve the solution. When no improvement occurs, such as a locally optimal solution is found, the hyper-heuristic tries to diversify the search to another region of the solution space by selecting appropriate heuristics from the given heuristic list.

Low-level heuristics usually represent the simple local search neighborhoods or the rules used by human experts for constructing solutions. However, more complex heuristics such as metaheuristics can also be included in the set of heuristics used by the hyper-heuristic [4]. Hyper-heuristics do not require knowledge of how each low-level heuristic works or the objective function of the problem. All domain-specific information remains at the level of the low-level heuristics and the objective function. For example, the hyper-heuristic only receives the value obtained as a result of evaluating the objective function.

In literature, two broad hyper-heuristic classes can be identified [8, 22]: heuristics to choose heuristics and heuristics to generate heuristics. In the first group, the hyper-heuristic is supplied with a set of low-level heuristics and the hyper-heuristic chooses how to apply these heuristics, e.g. in [5] and [20]. In the second group, good heuristics for a specific problem are evolved from given heuristic components, e.g. [6] and [21]. Both groups can further be divided into two classes: perturbative hyper-heuristics, e.g. as in [5], and constructive hyper-heuristics, e.g. as in [20]. Most of the approaches that fall into the first group work on a single candidate solution and decide which low level perturbative heuristic(s) will be applied at each iteration. In the second group of approaches, at each iteration, a solution is constructed from scratch by applying the constructive low-level heuristics. The hyper-heuristic determines the order in which these heuristics will be applied to generate the solution. In this study, we focus on a hyper-heuristic approach which is perturbative and falls in the group of heuristics to choose heuristics.

Perturbative heuristics used in a perturbative hyper-heuristic framework are divided into two classes [5]: mutational and hill climbing heuristics. Mutational heuristics modify a given solution randomly. In mutational heuristics, the resulting solution may be of lower quality than the current solution, however, hill climbing heuristics guarantee a solution which has better or the same quality as the input. Previous studies [5] show that choosing a low level mutational heuristic and then applying a single hill climbing heuristic generates better results than using the hill climber within the set of low level heuristics.

2.2 Landscape Analysis

A fitness landscape is obtained by associating a fitness value with each point in the search space, on which a neighborhood is defined. Then, a walk can be performed on this landscape by visiting neighboring points at each step. The definition of a neighborhood is crucial in the analysis of the fitness landscape and depends mainly on the solution representation and the heuristic operators used.

Fitness landscape analysis techniques are used to better understand the influence of representations and associated variation operators when solving a combinatorial optimization problem [3]. The concept of fitness landscape, which was introduced by Wright [1] to demonstrate the dynamics of biological evolutionary optimization, has been useful for the analysis and understanding of evolutionary algorithm's behavior. In addition, the study of fitness landscapes can be of value in designing heuristic search algorithms since it can help predict the algorithms' performance.

Ruggedness is commonly used to determine the structure of fitness landscapes and algorithm behavior. The auto-correlation function (acf_s), given in Equation 1, is used for analyzing the ruggedness of a landscape. By performing a random walk on a landscape, the ruggedness or the smoothness of the whole landscape may be described. The auto-correlation function is a measure of the correlation between the fitness of two points separated by s random steps. Auto-correlation function values close to 1 denote a high positive correlation and values close to 0 show low correlation.

$$acf_s = \frac{\sum_{t=1}^{T-s}\left(f_t - \bar{f}\right)\left(f_{t+s} - \bar{f}\right)}{\sum_{t=1}^{T}\left(f_t - \bar{f}\right)^2} \qquad (1)$$

The correlation length of a landscape, which is commonly used to compare the ruggedness of landscapes, is the largest number of steps, for which the starting and the end points of the walk are correlated. A high correlation length means a smoother landscape, whereas, a low correlation length shows a more rugged structure. Problems with smoother landscapes are easier for search algorithms which rely on the underlying neighborhood of the landscape to generate solutions. The correlation length (*cl*) is defined in Equation 2.

$$cl = -\frac{1}{\ln\left|acf_1\right|} \qquad (2)$$

In evolutionary algorithms literature, fitness landscape analysis has been studied for many years. Reeves, in [9], gives a detailed overview of the properties of landscapes. Other measures than *cl*, such as fitness distance correlation [14], epistasis variance [14], neutrality and evolvability [15] are also explored in some studies in literature. There are many application studies which perform a fitness landscape analysis of well known problems such as the traveling salesman problem [10], the multi-dimensional knapsack problem [3], the quadratic assignment problem [11], the graph bi-partitioning problem [12], the max-sat problem [13] and a special case of a routing problem [18].

In hyper-heuristics literature, there are only two studies [16,17], which focus on analyzing the landscape of hyper-heuristics. However, both of these studies work on constructive hyper-heuristics. In [16], a constructive hyper-heuristic which uses graph-coloring heuristics to solve the timetabling problem is considered. To analyze the heuristic landscape of this constructive hyper-heuristic, the fitness distance correlation and the correlation length metrics are used. In [17], a fitness distance correlation analysis is performed on the heuristic landscape of the hyper-heuristic which employs dispatching rules heuristics to solve the hybrid flow-shop problem. Both papers conclude that hyper-heuristics which exploit the properties of the heuristic landscape would be more successful for both problems.

3 Landscape Analysis of a Perturbative Hyper-heuristic Framework

Hyper-heuristic approaches search the heuristics space instead of the solution space. Therefore the corresponding landscape is called the "landscape of heuristics" and not that of the solutions. One of the most common methods to perform fitness landscape analysis is using an "auto correlation function". In our study, we adapt the "correlation length" (*cl*) metric which is derived from this function for analyzing the behavior of a subset of perturbative hyper-heuristics. The set of low-level perturbative heuristics, the heuristic selection method, the acceptance criteria and the hyper-heuristic framework are chosen from a previous study in [5] where a very detailed experimental comparison of perturbative hyper-heuristics is conducted. A generic hyper-heuristic framework managing a set of perturbative low level heuristics is chosen (Figure 1). In this framework, low-level heuristics are selected from a list, then the hyper-heuristic method applies these heuristics on the solution and finally the acceptance criteria decides whether the generated solution will be accepted or rejected.

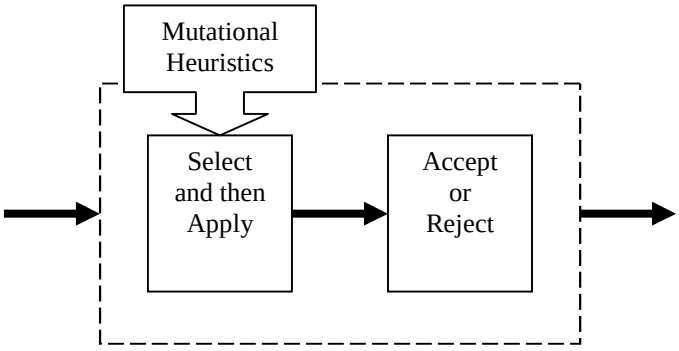

Figure 1. Illustration of a single iteration in a generic perturbative hyper-heuristic framework

The perturbative low-level heuristics which are used in this framework can be both mutational heuristics and hill climbers. However, in the preliminary experiments it was observed that when hill climber heuristics are used, a solution is found in a few iterations or search gets stuck at a local optimum. When conducting a landscape analysis, exploring different regions of the search space is more important than finding the optimum value of the problem. Therefore, only mutational heuristics are used in our experiments. Four different low-level mutational heuristics are used within the hyper-heuristics during the experiments as described below:

- Mutation (MUTN): This operator is the standard mutation operator used in Genetic Algorithms. All the bits are scanned one by one and a bit is flipped with a probability of *1*/*Solution_Length*.
- Swap Dimension (SWPD): Two dimensions within a given candidate solution are chosen randomly and then their values are exchanged.
- Dimensional Mutation (DIMM): A single dimension is chosen randomly. Then, mutation as in MUTN is applied only to the bits within the chosen dimension. A bit is flipped with a probability of *1/Dimension_Length*.
- Hyper-mutation (HYPM). This operator applies MUTN to all the bits of a given candidate solution with a probability of 0.5. HYPM heuristic helps the search to jump to new areas of the search space when a local optimum is encountered.

Greedy, Simple Random (SR) and Random Descent (RD) are the chosen heuristic selection methods. Greedy method allows all heuristics to process a given candidate solution and chooses the one that generates the most improved solution. SR heuristic selection mechanism randomly chooses a low level heuristic from the list with equal probability and applies it to the candidate solution. RD chooses a low level heuristic randomly with equal probability and applies the heuristic repeatedly until no improvement is achieved. All Moves (AM) and Only Improving (OI) schemes are chosen as the acceptance criteria. AM accepts all solutions generated by the selected heuristic, regardless of its quality. OI accepts only improving moves (solutions with equal quality are not accepted).

Landscape analysis is performed for the above defined hyper-heuristics on seven well known continuous benchmark functions: Sphere, Rosenbrock, Step, Quartic with (uniformly random) noise, Rastrigin, Schwefel and Griewangk (Table 1). Modality of a function specifies the number of optimum that it contains within the search space. *Unimodal* benchmark functions have a single optimum, whereas *multimodal* benchmark ones contain multiple optima. A function of *n* variables is referred to as *separable*, if it can be represented as a collection of *n* single-variable functions. For example, a multi-variable separable function can be equal to the sum of single variable functions. During the experiments, such *additively* separable benchmark functions are used, except the Griewangk function.

Table 1. Benchmark functions used in the experiments

Function		Range of x_i	Properties
Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$	[-5.12,5.12]	unimodal, separable
Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right)$	[-2.048,2.048]	unimodal, inseparable
Step	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n} \lfloor x_i \rfloor$	[-5.12,5.12]	unimodal, separable
Quartic with noise	$f(\vec{x}) = \sum_{i=1}^{n} \left(i \cdot x_i^4 + U(0,1) \right)$	[-1.28,1.28]	multimodal, separable
Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right)$	[-5.12,5.12]	multimodal, separable
Schwefel	$f(\vec{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} x_i \cdot \sin\left(\sqrt{\lvert x_i \rvert}\right)$	[-500,500]	multimodal, separable
Griewangk	$f(\vec{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600,600]	multimodal, inseparable

4 Experiments

4.1 Settings

In our experiments all benchmark functions are tested with 20 dimensions. We use a gray encoding to represent the solutions. In this scheme, each dimension is encoded with 10 bits. As a result, each candidate solution is a bit string of length 200. During each experimental run with a given hyper-heuristic over a benchmark function, the fitness value of an accepted solution is recorded at each step. A run (walk) is terminated after 2000 steps, sampling 2000 points (fitness values) for computing the correlation length. Moreover, the best fitness value encountered during a walk and the step that it is encountered are also accumulated. As each walk is repeated for 100 times, starting from different random point, statistics, such as, the average and standard error values for all these quantities are used during the landscape analysis.

4.2 Results and Discussion

Table 2 and 3 summarize the experimental results. In both tables, *cl* is the average correlation length obtained over 100 runs and *stderr(cl)* gives its standard error; *bs* is the average of the best solution obtained in each walk over 100 runs and the *stderr(bs)* gives its standard error; *step* is the average of the step count when the best solution is obtained in each walk over 100 runs and the *stderr(step)* gives its standard error. Each hyper-heuristic is denoted as <heuristic selection> – <move acceptance>. Table 2 shows the results obtained when the AM acceptance method is used together with the three heuristic selection schemes. Similarly, Table 3 shows the results obtained when the OI acceptance method is used together with the three heuristic selection schemes.

Table 2. Test result for All Moves acceptance criteria

Greedy – AM						
Function	*cl*	*stderr(cl)*	*bs*	*stderr(bs)*	*step*	*stderr(step)*
Sphere	44.83677	22.19067	0.00082	0.00341	1822.87	153.54758
Rosenbrock	15.05187	12.08595	67.84332	29.23171	1512.42	454.00527
Step	126.31290	46.58216	2.60000	1.42842	1731.01	199.89120
Quartic w/n	5.91090	1.81726	8.08736	0.57982	1271.22	488.93197
Rastigin	66.45556	28.08149	22.52137	7.03620	1958.86	53.54107
Schwefel	110.95565	40.78793	979.27836	322.72432	1960.16	69.05461
Griewank	51.36108	27.32004	1.26334	0.36940	1500.99	355.30855
RD – AM						
Function	*cl*	*stderr(cl)*	*bs*	*stderr(bs)*	*step*	*stderr(step)*
Sphere	3.27191	0.29425	66.07899	9.49872	1019.57	566.11031
Rosenbrock	2.93812	0.26570	1534.62562	314.93183	959.39	644.36727
Step	3.35915	0.29660	64.73000	4.49658	1056.03	580.98781
Quartic w/n	2.91038	0.22812	14.11436	1.48357	963.45	582.27345
Rastigin	3.31949	0.33136	207.03253	15.63024	942.77	557.72580
Schwefel	3.20222	0.28065	5204.39937	299.48174	1043.45	611.68927
Griewank	3.28610	0.32454	206.59255	27.57067	996.92	594.58138
SR – AM						
Function	*cl*	*stderr(cl)*	*bs*	*stderr(bs)*	*step*	*stderr(step)*
Sphere	3.40739	0.26373	77.52255	9.72065	994.28	526.58211
Rosenbrock	3.25867	0.31900	2167.60653	455.88516	986.82	543.36172
Step	3.44164	0.33630	69.87000	4.92089	907.72	519.35163
Quartic w/n	3.19732	0.26332	17.23813	2.07844	959.37	560.89696
Rastigin	3.30546	0.32932	233.21566	15.88990	1037.22	545.75616
Schwefel	3.24051	0.30786	5730.95627	289.34191	1047.42	607.88014
Griewank	3.31341	0.31405	238.64892	34.58293	1054.70	568.61063

When the AM selection technique is used, the Greedy hyper-heuristic has the highest correlation length values. This is mainly because the Greedy approach applies all the low-level heuristics and selects the best of them. Therefore, the new point is closely related with the previous point from which it was generated. However, the RD and the SR selection schemes might have high drops in fitness because of a bad heuristic selection. The intervals for the *cl* and *bs* values of the RD and SR methods are very close to each other. This means that, there is no statistically significant difference between them. Also RD behaves similar to SR and was not able to apply the chosen heuristic many times until there was no improvement.

In addition, since the Greedy approach always selects the best heuristic, it is able to find better solutions. The Greedy approach improves its solution at each iteration and thus is able to find better solutions, mostly towards

the end of the run. For example, for the Sphere function (which is unimodal), the Greedy – AM hyper-heuristic encounters its best solution around step 1823 on average. Due to the random heuristic selection mechanisms in both RD and SR, the best solution can be encountered at any step of the walk. Therefore, the standard error values of the step when the best solution is found are much higher than that of the Greedy method. Also, the solution quality achieved by RD and SR with AM acceptance is much lower than that of Greedy. These results back up the *cl* values obtained during the experiments. Higher *cl* means a smoother landscape which is easier for a search heuristic that relies on the neighborhood structure of the underlying landscape. In this experiment, the Greedy method which has a higher *cl* than the RD and SR, also achieves the best solution quality. The results obtained by RD and SR are similar to each other, so are the *cl* values of their landscapes.

Table 3. Test results for Only Improving acceptance criteria

Greedy – OI						
Function	*cl*	*stderr(cl)*	*bs*	*stderr(bs)*	*step*	*stderr(step)*
Sphere	49.51943	27.09491	0.00046	0.00110	1840.15	134.28932
Rosenbrock	15.38073	10.19749	45.12681	29.33253	1983.78	21.97132
Step	109.32785	36.54910	11.18000	2.69822	1720.08	238.57996
Quartic w/n	23.11167	19.32718	9.95526	1.37597	1300.77	515.69507
Rastigin	67.81996	22.53317	20.39235	6.11360	1938.68	105.83796
Schwefel	115.72481	42.782443	992.92127	383.29499	1964.38	50.71882
Griewank	46.19313	24.16803	0.70176	0.69001	1638.47	256.44840
RD – OI						
Function	*cl*	*stderr(cl)*	*bs*	*stderr(bs)*	*step*	*stderr(step)*
Sphere	73.08927	41.64527	0.04087	0.06379	1961.78	42.75383
Rosenbrock	24.50513	15.90997	51.54439	28.78878	1982.84	16.73363
Step	167.24141	65.73952	14.03000	3.00993	1692.28	277.38268
Quartic w/n	33.88447	27.85375	10.25615	1.36049	1297.24	555.59039
Rastigin	98.80451	36.87493	29.80780	7.51286	942.77	557.72580
Schwefel	136.34661	59.51040	1583.14879	447.80914	1964.53	43.42285
Griewank	68.06959	31.32192	1.17834	0.76097	1899.97	110.48967
SR – OI						
Function	*cl*	*stderr(cl)*	*bs*	*stderr(bs)*	*step*	*stderr(step)*
Sphere	159.05777	100.41520	2.65562	2.54339	1980.06	22.02286
Rosenbrock	51.81645	35.3206	87.21807	35.79628	1977.99	21.35202
Step	315.74646	166.49767	21.54000	4.23887	1804.17	175.50147
Quartic w/n	52.44623	44.92910	11.52074	1.80997	1453.03	495.61068
Rastigin	202.77516	88.83062	47.91345	11.82588	1959.82	37.29526
Schwefel	219.58939	103.99239	2235.31564	444.1147	1968.58	29.94445
Griewank	148.47994	92.03618	7.93971	6.80547	1982.67	16.06870

If the appropriate heuristic is not selected, this might cause a large decrease in the fitness value, which makes the landscape more rugged. When the OI acceptance is used, these fitness decreases are bounded, as OI allows only improving solutions to be accepted at any step of the search. Therefore *cl* values of almost all functions are increased when OI acceptance is used. In addition to this, since worse results are eliminated, *bs* values are also better. For RD and SR hyper-heuristics, the increase in *cl* values is much more than that of Greedy, since in Greedy the best heuristic is always chosen. This means that in most cases the new solution is not much worse than the current one. Therefore the improvement in *cl* and solution quality for Greedy with OI is less. With SR, the average *cl* is increased dramatically; however, the standard error is very high which is due to the randomness in the heuristic selection process. RD seems to benefit the most with regard to solution quality, from using OI instead of AM. Even though the average increase in *cl* for RD is less than that of SR, the standard error is smaller. Therefore it is able to find better solutions in more cases than SR, thus increasing solution quality. Again, the results show that the *cl* values and solution quality are in accordance.

5. Conclusion and Future Work

In this study, we performed a correlation length analysis on the heuristic landscapes generated by six hyper-heuristics on seven commonly used benchmark functions. The heuristic selection mechanisms used are the Greedy method, the Simple Random and the Random Descent methods. As acceptance schemes, All Moves and Only Improving are used. The results of the experiments show that the heuristic selection mechanism as well as the solution acceptance scheme affects the underlying heuristic landscape. We also showed that as the

correlation length increases, solution quality of the hyper-heuristics also increases, as expected. As a conclusion, we observe that the correlation length analysis of the heuristic landscape, produced by a hyper-heuristic provides a good indication of the algorithm performance.

We experimented with a subset of existing heuristic selection and acceptance criteria combinations in a generic hyper-heuristic framework, referred to as type A in [5]. The results of the preliminary investigations are very promising which promote further study. As a future work, other perturbative hyper-heuristic frameworks, namely type B, C and D as described in [5], embedding more sophisticated heuristic selection schemes (i.e., Choice Function, Tabu Search) and acceptance mechanisms (i.e., Great Deluge, Simulated Annealing) will be analyzed on a more diverse set of benchmark functions.

References:

[1] S. Wright, "The roles of mutation, inbreeding, crossbreeding and selection in evolution", in *Proc. VI Int. Conf. Genetics*, vol. 1, pp. 356–366, 1932.

[2] P.Cowling, G. Kendall, E. Soubeiga, "A hyper-heuristic approach to scheduling a sales summit", in *PATAT 2000, LNCS*, vol. 2079, pp. 176–190, Springer, 2000.

[3] J. Tavares, F. B. Perreria, E.Costa, "Multidimensional Knapsack Problem: A Fitness Landscape Analysis", *IEEE Transactions on Systems, Man and Cybernetics – Part B.* Vol. 38, No. 3, 2008.

[4] K. Chakhlevitch, P. Cowling, "Hyper-heuristics: Recent Developments", *Adaptive and Multilevel Metaheuristics*, SCI 136, pp. 3–29, 2008.

[5] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A Comprehensive Survey of Hyper-heuristics", *Intelligent Data Analysis*, 12(1):1-21, 2008.

[6] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring Hyper-heuristic methodologies with genetic programming", *Studies in Computational Intelligence: collaboration, fusion and emergence*, chapter 6, Springer, 2009.

[7] E. Özcan, S. Etaner-Uyar, E. K. Burke, "A Greedy Hyper-heuristic in Dynamic Environments", *GECCO 2009 Workshop on Automated Heuristic Design: Crossing the Chasm for Search Methods*, 2009.

[8] E. Soubeiga, "Development and Application of Hyper-heuristics to Personnel Scheduling", *PhD Thesis in School of Computer Science and Information Technology The University of Nottingham*, 2003.

[9] C. R. Reeves, "Fitness Landscapes and Evolutionary Algorithms", in Proc. Sel. Papers 4th Eur. Conf. Artificial Evolution, LNCS, vol. 1829: pp. 3-20, 1999.

[10] P. F. Stadler, W. Schnabl, "The Landscape of the Traveling Salesman Problem", Physics Letters A, vol. 161: pp. 337-344, 1992.

[11] P. Merz, B. Freisleben, "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem", IEEE Trans. Evol. Comp., vol 4: No.4, 2000.

[12] E. Angel, V. Zissimopoulos, "Auto-correlation Coefficient for the Graph Bipartitioning Problem", Theoretical Computer Science, vol. 191: pp. 229-243, 1998.

[13] H. H. Hoos, K. Smyth, T. Stützle, "Search Space Features Underlying the Performance of Stochastic Local Search Algorithms for MAX-SAT", Parallel Problem Solving from Nature - PPSN VIII, LNCS vol. 3242: pp. 51-60, 2004.

[14] B. Naudts, L. Kallel, "Comparison of Summary Statistics of Fitness Landscapes", IEEE Trans. Evol. Comp. vol.41: pp. 1-15, 2000.

[15] T. Smith, P. Husbands, P. Layzell, M. O'Shea, "Fitness landscapes and evolvability", Evolutionary Computation, vol. 10: No. 1, pp. 1-34, 2002.

[16] G. Ochoa, R. Qu, E. K. Burke, "Analyzing the Landscape of a Graph Based Hyper-heuristics for Timetabling Problems", Genetic and Evolutionary Computation Conference (GECCO), 2009.

[17] G. Ochoa, J. A. Vazquez-Rodriguez, S. Petrovic, E. K. Burke, "Dispatching Rules for Production Scheduling: a Hyper-heuristic Landscape Analysis", IEEE Congress on Evolutionary Computation (CEC), 2009.

[18] F. Corut-Ergin, A. Yayimli., A. S. Uyar, "An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks", EvoCOMNET 2009: Sixth European Workshop on the Application of Nature-inspired Techniques for Telecommunication Networks and other Parallel and Distributed Systems, LNCS, vol. 5484, pp.31-40, Springer, 2009.

[19] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, S. Schulenburg, "Hyper-heuristics: An Emerging Direction in Modern Search Technology", Handbook of Metaheuristics, pp. 457-474, Kluwer, 2003.

[20] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, "A Graph-based Hyper-heuristic for Educational Timetabling Problems", Europena Journal of Operational Research, vol.176, pp. 177-192, 2007.

[21] M. Bader-El-Den, R. Poli, "Generating SAT Local-Search Heuristics using a GP Hyper-heuristic Framework, 8th International Conference on Artificial Evolution (EA), LNCS, vol. 4926, pp. 37-49, Springer, 2008.

[22] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "A Survey of Hyper-heuristics", technical report, University of Nottingham, 2009.