An Improved Choice Function Heuristic Selection for Cross Domain Heuristic Search

John H. Drake[1], Ender Özcan[1], and Edmund K. Burke[2]

[1] School of Computer Science, University of Nottingham
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{jqd,exo}@cs.nott.ac.uk

[2] Computing Science and Mathematics, School of Natural Sciences
University of Stirling, Stirling, FK9 4LA, Scotland
e.k.burke@stir.ac.uk

Abstract. Hyper-heuristics are a class of high-level search technologies to solve computationally difficult problems which operate on a search space of low-level heuristics rather than solutions directly. A iterative selection hyper-heuristic framework based on single-point search relies on two key components, a heuristic selection method and a move acceptance criteria. The Choice Function is an elegant heuristic selection method which scores heuristics based on a combination of three different measures and applies the heuristic with the highest rank at each given step. Each measure is weighted appropriately to provide balance between intensification and diversification during the heuristic search process. Choosing the right parameter values to weight these measures is not a trivial process and a small number of methods have been proposed in the literature. In this study we describe a new method, inspired by reinforcement learning, which controls these parameters automatically. The proposed method is tested and compared to previous approaches over a standard benchmark across six problem domains.

Keywords: Hyper-heuristics, Choice Function, Heuristic Selection, Cross-domain Optimisation, Combinatorial Optimization.

1 Introduction

The term 'hyper-heuristic' was first used in the field of combinatorial optimisation by Cowling et al. [1] and was defined as '*heuristics to choose heuristics*'. This paper investigated the application of a number of random, greedy and Choice Function-based hyper-heuristic approaches to a real-world sales summit scheduling problem using two deterministic move acceptance criteria, all moves (AM) and only improving (OI). Although the term hyper-heuristic was first used at this time, ideas which exhibited hyper-heuristic behaviour can be traced back as early as 1961 [2] in the field of job shop scheduling where combining scheduling rules was shown to perform better than taking any of the rules individually. In the first journal article to appear using the term Burke et al. [3] presented a tabu-search-based hyper-heuristic. In this system a set of low-level heuristics are ranked using rules based on reinforcement learning and compete against each other for selection. The hyper-heuristic selects the highest ranked heuristic not present in the tabu list. If an improvement is made after applying the selected heuristic its rank is increased, if not, its rank is decreased and it is placed in the tabu list until the current solution has changed. This hyper-heuristic was applied to nurse scheduling and university course timetabling problems obtaining competitive results. Hyper-heuristics have since been applied successfully to a wide range of problems such as examination timetabling [3–7], production scheduling [2], nurse scheduling [3, 8], bin packing [8, 9], sports scheduling [10], dynamic environments [11] and vehicle routing [8, 12].

Research trends have lead to a number of different hyper-heuristics approaches being developed, particularly those concerned with automatically generating new heuristics, for which the original definition of a hyper-heuristic is too limited to cover. A more general definition is offered by Burke et al. [13, 14]:

> 'A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational search problems.'

This more general terminology includes systems which use high level strategies other than heuristics within the definition of hyper-heuristics and covers the two main classes of hyper-heuristics, those concerned with heuristic selection and those with heuristic generation. Here, our concern will be those methodologies which are used to select heuristics.

2 Selection Hyper-Heuristics and the Choice Function

Traditional single-point based search hyper-heuristics rely on two key components, a heuristic selection method and a move acceptance criteria as decomposed by Özcan et al. [15] and depicted in Figure 1. Such hyper-heuristics will sometimes be labelled *selection method-acceptance criteria* in this paper. Hyper-heuristics using this framework operate on a single solution and repeatedly select and apply low-level heuristics to this solution. At each stage a decision made as to whether to accept the move until some termination criteria is met.

Cowling et al. [1] experimented with a number of heuristic selection mechanisms including Simple Random and Choice Function using accept All Moves and accept Only Improving moves as acceptance criteria. Simple Random selects a heuristic to apply randomly from the set of low-level heuristics at each point in the search. The Choice Function is an elegant selection method which scores heuristics based on a combination of three different measures. The heuristic to apply is then be chosen by a strategy based on these scores. The first measure (f_1) records the previous performance of each individual heuristic, with more recent executions carrying larger weight. The value of f_1 for each low-level heuristic $h_1, h_2, ..., h_j$ is calculated as:

$$f_1(h_j) = \sum_n \alpha^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \tag{1}$$

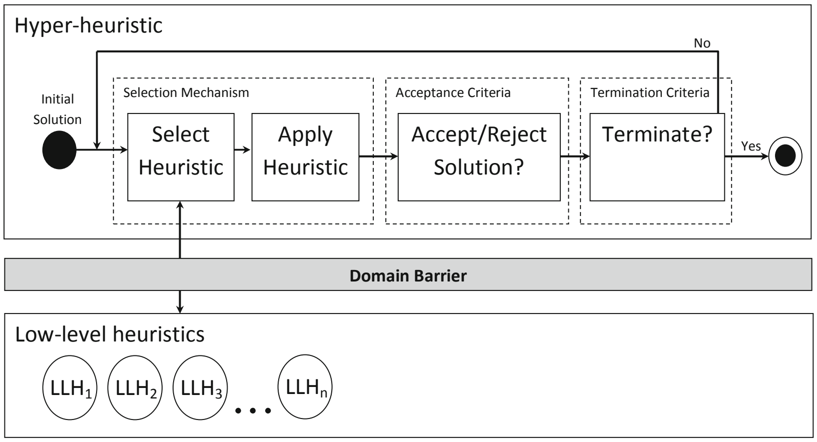

Fig. 1. Classic single-point search hyper-heuristic framework

where $I_n(h_j)$ is the change in evaluation function, $T_n(h_j)$ is the time taken to call the heuristic for each previous invocation n of heuristic h_j and α is a value between 0 and 1 giving greater importance to recent performance.

The second measure (f_2) attempts to capture any pair-wise dependencies between heuristics. Values of f_2 are calculated for each heuristic h_j when invoked immediately following h_k using the formula in Equation 6:

$$f_2(h_k, h_j) = \sum_n \beta^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)} \tag{2}$$

where $I_n(h_k, h_j)$ is the change in evaluation function, $T_n(h_k, h_j)$ is the time taken to call the heuristic for each previous invocation n of heuristic h_j following h_k and β is a value between 0 and 1 which also gives greater importance to recent performance.

The third measure (f_3) is the time elapsed ($\tau(h_j)$) since the heuristic was last selected by the Choice Function. This allows all heuristics at least a small chance of selection.

$$f_3(h_j) = \tau(h_j) \tag{3}$$

In order to rank heuristics a score is given to each heuristic with Choice Function F calculated as:

$$F(h_j) = \alpha f_1(h_j) + \beta f_2(h_k, h_j) + \delta f_3(h_j) \tag{4}$$

where α and β as defined previously weight f_1 and f_2 respectively to provide intensification of the heuristic search process whilst δ weights f_3 to provide sufficient diversification. In this initial work these parameters were set as static values based on the authors experimental insight. This study showed the Choice Function selection combined with All Moves acceptance worked well.

Further to this work, Cowling et al. [16] described a method to adaptively change these parameters. A mechanism is proposed which increases the weights of α or β when using a heuristic selected by the Choice Function results in an improvement in the objective value. Although no specific implementation details are provided, this reward is said to be proportional to the size of improvement over the previous solution. Conversely, if a decrease in solution quality is obtained these weights are penalised proportionally to the change in objective value. Using this mechanism lead to an improved performance compared to the original results. Here we will use the implementation of the Choice Function used by Bilgin et al. [17] for benchmark function optimisation and Özcan et al. [4] and Burke et al. [6] for Examination Timetabling. This implementation increases α and β and reduces δ by the same value if an improvement is made and reduces α and β and increases δ if no improvement is made.

3 Modified Choice Function

Recently the HyFlex framework [8] was proposed and developed in order to support the first Cross-domain Heuristic Search Challenge, CHeSC 2011 [18]. HyFlex was designed with the goal of providing a common framework to test and compare different cross domain algorithms. Currently HyFlex contains six problem domains for algorithms to be tested on; maximum satisfiability (MAX-SAT), one-dimensional bin packing, personnel scheduling, permutation flow shop, the travelling salesman problem (TSP) and the vehicle routing problem (VRP). Using this framework allows us to directly compare our approach with previously proposed algorithms.

Using the classic version of the Choice Function has some limitations when applied to the HyFlex framework. Firstly, we are often not interested in the proportional improvement gained by a given heuristic but rather whether there has been any improvement at all. In the early stages of a search, a relatively poor heuristic could gain a large reward if it obtains a large improvement in objective value from a poor starting position. Later on in the search, a heuristic may yield a small improvement which is much more significant in the context of the optimisation process but will not receive such a large reward for this improvement. Secondly, if no improving solutions are found for a period of time, the Choice Function can very quickly descend into random search if the weighting is dominated by the diversification component. This can be a useful trait however the rate at which the diversification increases in significance must be controlled. Özcan et al. [4] observed that Simple Random heuristic selection with Late Acceptance Strategy move acceptance performed very well on a set of Examination Timetabling instances. In this particular case very few (4) perturbative low-level heuristics were implemented.

We propose a modified version of the Choice Function which aims to address these issues through the management of the parameters weighting f_1, f_2 and f_3 inspired by reinforcement learning [19]. This mechanism will rely on a system of reward and punishment in order to tune these parameters. Our Modified Choice

Function does not make a distinction between the values of α or β which weight f_1 and f_2 respectively and considers them as a single intensification parameter which we will refer to as simply ϕ. This value will also be used to give greater importance to recent performance as with the original Choice Function of Cowling et al. [1]. The parameter to weight f_3 is used to control the level of diversification of heuristic search as before and will still be referred to as δ. In the Modified Choice Function the score F_t for each heuristic h_j is now calculated as:

$$F_t(h_j) = \phi_t f_1(h_j) + \phi_t f_2(h_k, h_j) + \delta_t f_3(h_j) \tag{5}$$

where t is the number of invocations of h_j since an improvement was made using this heuristic. At each stage, if an improvement in objective value is made ϕ is rewarded and set to a static maximum value close to the upper limit of the interval (0,1) whilst δ is concurrently reduced to a static minimum value close to the bottom end of this interval. This leads to a greater emphasis on intensification and greatly reduces the level of diversity of heuristic selection choice each time an improvement is obtained. If no improvement in objective value is made the level of intensification is decreased by linearly reducing ϕ and the weighting of diversification is increased at the same rate. This gives the intensification component of the Choice Function more time as the dominating factor in the calculation of F. For the experiments in this paper we define the parameters ϕ_t and δ_t as:

$$\phi_t(h_j) \; = \begin{cases} 0.99, & \text{if an improving move is made} \\ \max\{\phi_{t-1} - 0.01, 0.01\}, & \text{if a non-improving move is made} \end{cases} \tag{6}$$

and

$$\delta_t(h_j) \; = 1 - \phi_t(h_j) \tag{7}$$

4 Computational Results

Prior to the original competition, the results of eight hyper-heuristics were provided by the organisers to assess an algorithms performance [18]. These hyper-heuristics were inspired by state-of-the-art techniques from the hyper-heuristic literature. Each hyper-heuristic performs a single run on 10 instances for each of 4 problem domains; maximum satisfiability (MAX-SAT), one-dimensional bin packing, personnel scheduling and permutation flow shop. They are then ranked using a system based on the Formula One scoring system, the best performing hyper-heuristic for each instance is awarded 10 points, the second 8 points and then each further hyper-heuristic awarded 6, 5, 4, 3, 2, 1 and 0 points respectively. As this ranking system is based on relative performance, the Choice Function and Modified Choice Function are compared to the competition entries independently. All experiments were carried out on machines allowing a hyper-heuristic 576 seconds running time for each instance by the benchmarking tool provided by the competition organisers. In order for a fair comparison to be made crossover heuristics are ignored as the original Choice Function provides no details of how to manage operators which require more than one argument. Figure 2(a) shows the results of the Modified Choice Function using accept All Moves as an acceptance criteria when compared with the eight hyper-heuristics (HH1-HH8) provided for the competition. Figure 2(b) shows the results of the same experiments using the original Choice Function and accept All Moves acceptance as implemented by Bilgin et al. [17] and Burke et al. [6].

	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	ModCF-AM
MAX-SAT	55.25	73.25	36.5	24.5	1	46	51.75	14.5	**87.25**
Bin Packing	59	61	**76**	71	15	51	39	1	17
Personnel Scheduling	64	57.5	22	50.5	50	0	49.5	31	**65.5**
Flow Shop	30	21	26.5	86	19.5	**77.5**	21	69	39.5
Overall	208.25	212.75	161	**232**	85.5	174.5	161.25	115.5	209.25

(a)

	HH1	HH2	HH3	HH4	HH5	HH6	HH7	HH8	CF-AM
MAX-SAT	58	**78**	39.5	25.5	1	49	54.5	15.5	69
Bin Packing	59	61	**78**	71	19	51	38	7	6
Personnel Scheduling	**65.5**	64.5	23	52.5	53	0	52	31	48.5
Flow Shop	38	27.5	32	**86**	25.5	78.5	26.5	70	6
Overall	220.5	231	172.5	**235**	98.5	178.5	171	123.5	129.5

(b)

Fig. 2. Formula One scores for a single run of Modified Choice Function - All Moves hyper-heuristic and the CHeSC default hyper-heuristics (a) and a single run of classic Choice Function - All Moves hyper-heuristic and the CHeSC default hyper-heuristics

From these tables we see that the Modified Choice Function outperforms all of the CHeSC default hyper-heuristics in MAX-SAT and Personnel Scheduling. More importantly the Modified Choice Function outperforms the original Choice Function in all four problem domains although both versions seem to struggle more on the Bin Packing and Flow Shop instances. This could be due to the omission of crossover operators if such operators perform well in these problem domain. The best performing hyper-heuristic in this set (HH4) is based on iterative local search, this supports the work of Özcan et al. [20] which showed that the F_C selection hyper-heuristic framework performed well compared to other hyper-heuristic frameworks.

Following the competition the results were provided for the competition entries over a subset of the problems of all six problem domains. These results were taken as the median of 31 runs of each hyper-heuristic on each instance. Our results are also taken as the median of 31 runs in order to maintain consistency and allow direct comparison to the competition entries. Figure 3(a) shows the results of the classic Choice Function and All Moves acceptance criteria compared to the 20 competition entries using the Formula One scoring system. Figure 3(b) shows the results of the same experiments using the Modified Choice Function and accept All Moves as an acceptance criteria.

Rank	Name	Score
1	AdapHH	181
2	VNS-TW	134
3	ML	131.5
4	PHunter	93.25
5	EPH	89.25
6	HAHA	75.75
7	NAHH	75
8	ISEA	71
9	KSATS-HH	66
10	HAEA	53.5
11	ACO-HH	39
12	GenHive	36.5
13	DynILS	27
14	SA-ILS	24.25
15	XCJ	22.5
16	AVEG-Nep	21
17	GISS	16.75
18	SelfSearch	7
19	MCHH-S	4.75
20	**Classic CF - AM**	**1**
21	Ant-Q	0

(a)

Rank	Name	Score
1	AdapHH	177.1
2	VNS-TW	131.6
3	ML	127.5
4	PHunter	90.25
5	EPH	88.75
6	NAHH	72.5
7	HAHA	71.85
8	ISEA	68.5
9	KSATS-HH	61.35
10	HAEA	52
11	ACO-HH	39
12	**Modified CF - AM**	**38.85**
13	GenHive	36.5
14	DynILS	27
15	SA-ILS	22.75
16	XCJ	20.5
17	AVEG-Nep	19.5
18	GISS	16.25
19	SelfSearch	5
20	MCHH-S	3.25
21	Ant-Q	0

(b)

Fig. 3. Results of the median of 31 runs of the classic Choice Function - All Moves hyper-heuristic (a) and the Modified Choice Function - All Moves hyper-heuristic (b), compared to CHeSC competitors using Formula One scores

Since the competition results were made available, Di Gaspero and Urli [21] described variations of their original method (AVEG-Nep), which are also based on Reinforcement Learning. The best of the variants included in this paper ranked 13th overall compared to the original competitors. Here we see that managing the parameter settings of a Choice Function using Reinforcement Learning inspired techniques can outperform such methods, ranking 12th overall. For this hyper-heuristic points are only scored in two problem domains, Personnel Scheduling and MAX-SAT leaving room for improvement in the other four domains. The vast majority (32.85) of these points were scored in MAX-SAT where our method excels. When compared to the competition entries, the Modified Choice Function outperforms all other competitors. It is likely that a very small number of heuristics are providing improvement in this problem domain and the increased focus on intensification is providing the gain in performance. Figure 4 shows a breakdown of the number of points awarded to each technique over the MAX-SAT competition instances.

Using the classic choice function performs particularly badly against the other competition entries ranking 20th out of 21 overall only obtaining a single point in Personnel Scheduling. The Formula One scoring system is limited in that

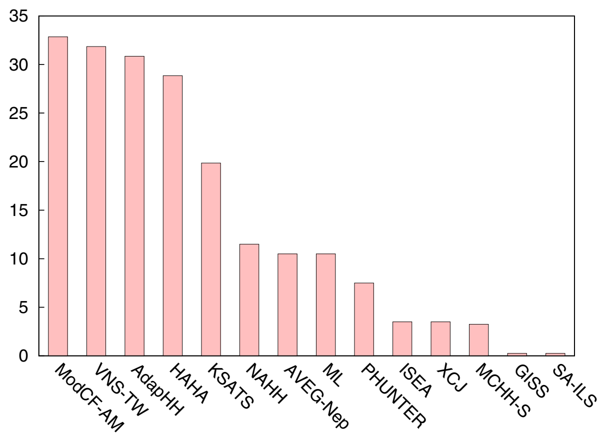

Fig. 4. Number of points scored in the MAX-SAT domain using the Formula One system for each CHeSC competitor

it only measures relative performance against a set of previous approaches. As such a more direct comparison between the two experiments can be performed on the objective values achieved by both hyper-heuristics. Table 1 shows the results of an independent student's t-test on the values of each of the 31 runs for each instance in the competition within a 95% confidence interval. These results show the Modified Choice Function statistically significantly outperforming the classic Choice Function completely in 3 of the 6 problem domains. In many cases there is no statistically significant difference in performance. In only 3 of the 30 problem instances the classic choice function performs statistically significantly better than the Modified Choice Function.

Table 1. Pairwise comparison between MCF-AM and CF-AM using independent T-Test. In this table s+ (s-) denotes that using MCF-AM (CF-AM) is performing statistically significantly better than using CF-AM (MCF-AM), while =+ (=−) denotes that there is no statistically significant performance variation between MCF-AM and CF-AM however MCF-AM (CF-AM) performs slightly better (worse) on average.

Problem Domain	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
MAX-SAT	s+	s+	s+	s+	s+
Bin Packing	=−	s+	s+	=−	s−
Personnel Scheduling	=−	=+	=+	=−	=+
Flow Shop	s+	s+	s+	s+	s+
TSP	s−	=−	s−	s+	=+
VRP	s+	s+	s+	s+	s+

5 Concluding Remarks

In this work we have described a modified version of the Choice Function heuristic selection method which manages the parameters which weight the intensification and diversification components of Choice Function scores through methods inspired by reinforcement learning. This Modified Choice Function aggressively rewards the intensification weighting and heavily punishes the diversification component each time an improvement is made. We have shown that managing these parameters in such a way provides great benefits compared to a classic implementation of the Choice Function. So far, this work has been limited to improving the Choice Function selection mechanism itself. Previous work in the literature has suggested that performance can be improved by reducing the search space of heuristics [22, 23]. We plan to include the method proposed by Özcan and Kheiri [24] to reduce the set of active heuristics in combination with the Modified Choice Function heuristic selection method and apply it to the problem instances available in HyFlex. We restricted this study to focus on only the selection mechanism component of a traditional hyper-heuristic, Özcan et al. [15, 20] tested a number of hyper-heuristics over a set of benchmark functions and observed that the acceptance criteria used can have a more significant impact on the performance of a hyper-heuristic than selection mechanism. We would like to extend this work to analyse the effect of using different move acceptance criteria in conjunction with the Modified Choice Function. In this paper we have not made use of any operators in the HyFlex framework which require more than one argument such as crossover. Drake et al. [25] described a number of methods for managing potential second arguments for crossover and other n-ary operators. As future work we will include the multiple argument management techniques from this study to analyse whether including crossover operators can benefit hyper-heuristics based on the Modified Choice Function.

References

1. Cowling, P.I., Kendall, G., Soubeiga, E.: A Hyperheuristic Approach to Scheduling a Sales Summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
2. Fisher, M., Thompson, G.: Probabilistic learning combinations of local job-shop scheduling rules. In: Factory Scheduling Conference (1961)
3. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9, 451–470 (2003)
4. Özcan, E., Bykov, Y., Birben, M., Burke, E.K.: Examination timetabling using late acceptance hyper-heuristics. In: CEC 2009, pp. 997–1004 (2009)
5. Özcan, E., Misir, M., Ochoa, G., Burke, E.K.: A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. Int. Journal of Applied Metaheuristic Computing 1, 39–59 (2010)
6. Burke, E.K., Kendall, G., Misir, M., Özcan, E.: Monte carlo hyper-heuristics for examination timetabling. Annals of Operations Research (in press)
7. Sabar, N.R., Ayob, M., Qu, R., Kendall, G.: A graph coloring constructive hyper-heuristic for examination timetabling problems. Applied Intelligence (in press)
8. Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J.A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A.J., Petrovic, S., Burke, E.K.: HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 136–147. Springer, Heidelberg (2012)

9. López-Camacho, E., Terashima-Marín, H., Ross, P.: A hyper-heuristic for solving one and two-dimensional bin packing problems. In: GECCO 2011, pp. 257–258. ACM, New York (2011)
10. Gibbs, J., Kendall, G., Özcan, E.: Scheduling English Football Fixtures over the Holiday Period Using Hyper-heuristics. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 496–505. Springer, Heidelberg (2010)
11. Kiraz, B., Uyar, A.Ş., Özcan, E.: An Investigation of Selection Hyper-heuristics in Dynamic Environments. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A.I., Merelo, J.J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 314–323. Springer, Heidelberg (2011)
12. Garrido, P., Castro, C.: Stable solving of cvrps using hyperheuristics. In: GECCO 2009, pp. 255–262. ACM (2009)
13. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.: A Classification of Hyper-heuristics Approaches. In: Handbook of Metaheuristics, 2nd edn., pp. 449–468. Springer (2010)
14. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. Technical Report No. NOTTCS-TR-SUB-0906241418-2747, School of Comp. Sci., University of Nottingham (2010)
15. Özcan, E., Bilgin, B., Korkmaz, E.E.: Hill Climbers and Mutational Heuristics in Hyperheuristics. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN IX. LNCS, vol. 4193, pp. 202–211. Springer, Heidelberg (2006)
16. Cowling, P., Kendall, G., Soubeiga, E.: A parameter-free hyperheuristic for scheduling a sales summit. In: MIC 2001, pp. 127–131 (2001)
17. Bilgin, B., Özcan, E., Korkmaz, E.E.: An Experimental Study on Hyper-Heuristics and Exam Timetabling. In: Burke, E.K., Rudová, H. (eds.) PATAT 2006. LNCS, vol. 3867, pp. 394–412. Springer, Heidelberg (2007)
18. Ochoa, G., Hyde, M.: The cross-domain heuristic search challenge (CHeSC 2011) (2011), `http://www.asap.cs.nott.ac.uk/chesc2011/`
19. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
20. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12(1), 3–23 (2008)
21. Di Gaspero, L., Urli, T.: Evaluation of a family of reinforcement learning cross-domain heuristics for optimization. In: LION 6 (2012)
22. Özcan, E., Basaran, C.: A case study of memetic algorithms for constraint optimization. Soft Computing 13(8-9), 871–882 (2009)
23. Chakhlevitch, K., Cowling, P.I.: Choosing the Fittest Subset of Low Level Heuristics in a Hyperheuristic Framework. In: Raidl, G.R., Gottlieb, J. (eds.) EvoCOP 2005. LNCS, vol. 3448, pp. 23–33. Springer, Heidelberg (2005)
24. Özcan, E., Kheiri, A.: A hyper-heuristic based on random gradient, greedy and dominance. In: ISCIS 2011, pp. 404–409 (2011)
25. Drake, J.H., Özcan, E., Burke, E.K.: Controlling crossover in a selection hyper-heuristic framework. Technical Report No. NOTTCS-TR-SUB-1104181638-4244, School of Computer Science, University of Nottingham (2011)