An Ant-based Selection Hyper-heuristic for Dynamic Environments

Berna Kiraz[1], A. Şima Etaner-Uyar[2], and Ender Özcan[3]

[1] Institute of Science and Technology, Istanbul Technical University, Turkey
Berna.Kiraz@marmara.edu.tr
[2] Department of Computer Engineering, Istanbul Technical University, Turkey
etaner@itu.edu.tr
[3] School of Computer Science, University of Nottingham, UK
Ender.Ozcan@nottingham.ac.uk

Abstract. Dynamic environment problems require adaptive solution methodologies which can deal with the changes in the environment during the solution process for a given problem. A selection hyper-heuristic manages a set of low level heuristics (operators) and decides which one to apply at each iterative step. Recent studies show that selection hyper-heuristic methodologies are indeed suitable for solving dynamic environment problems with their ability of tracking the change dynamics in a given environment. The choice function based selection hyper-heuristic is reported to be the best hyper-heuristic on a set of benchmark problems. In this study, we investigate the performance of a new learning hyper-heuristic and its variants which are inspired from the ant colony optimisation algorithm components. The proposed hyper-heuristic maintains a matrix of pheromone intensities (utility values) between all pairs of low level heuristics. A heuristic is selected based on the utility values between the previously invoked heuristic and each heuristic from the set of low level heuristics. The ant-based hyper-heuristic performs better than the choice function and even its improved version across a variety of dynamic environments produced by the Moving Peaks Benchmark generator.

1 Introduction

Many real world constraint optimisation problems contain a set of components which might change in time separately or concurrently. Some of these components include the problem instance, the objectives and the constraints. A *good* solution method needs to be adaptive and intelligent to be able to deal with the complexities introduced by such a dynamic environment and track the changes. Branke [2] categorized these changes based on (i) *frequency*, (ii) *severity*, (iii) *predictability* of a change, and (iv) *cycle length/cycle accuracy* which is a property defining somewhat periodicity and precision of changes. There is a variety of approaches dealing with different types of changes in literature. Most of the

approaches handling dynamic environment problems are modified from the existing approaches designed for solving static problems based on different strategies. More details on dynamic environments can be found in [2,5,11,16].

In this study, we use an emerging search methodology, namely *selection hyper-heuristics* for solving dynamic environment problems. A selection hyper-heuristic controls a set of low level heuristics, choosing the most appropriate one to apply to a solution in hand and deciding to accept or reject the newly created solution at each step [3]. There are learning hyper-heuristic methodologies as well as the ones which do not utilise any learning at all. An online learning selection hyper-heuristic can incorporate learning either into the heuristic selection, or move acceptance methods. These components attempt to improve performance during the search process. If learning takes place during heuristic selection, frequently, a mechanism is used to score the performance of each low level heuristic. Then a heuristic is chosen based on these scores. Nareyek [12] tested *Reinforcement Learning* (RL) heuristic selection on Orc Quest and modified logistics domain. Initially, scores are initialised to the same value for all heuristics, e.g., 0. After a chosen heuristic is applied to a solution, if there is improvement, then its score is increased; otherwise it is decreased, e.g. by one. A heuristic is chosen using different mechanisms and the best strategy appears to be selecting the heuristic with maximum score. Cowling et al. [4] investigated the performance of different heuristic selection methods on a real world scheduling problem. One of them is a learning method referred to as *Choice Function* (CF) which maintains a score for each low level heuristic taking a weighted average of three values; how well it performs individually and as a successor of the previously invoked heuristic and the elapsed time since its last call. Then the heuristic with the maximum score is selected and applied to the current solution at a given step. Drake and Özcan [7] proposed a modified version of CF improving its performance (ICF) in which weights dynamically change, enforcing the search process to go into diversification faster than usual, when the successive moves are non-improving.

There is strong empirical evidence that selection hyper-heuristics work for not only discrete combinatorial problems [3] but also discrete and continuous dynamic environment problems [9,10,15,14], being able to respond to the changes in such an environment rapidly. In this study, we describe a new learning hyper-heuristic for dynamic environments, which is designed based on the ant colony optimisation algorithm components. The proposed hyper-heuristic maintains a matrix of pheromone intensities (utility values) between all pairs of low level heuristics. A heuristic is selected based on the utility values between the previously invoked heuristic and each heuristic from the set of low level heuristics. We investigate the performance of the proposed hyper-heuristic controlling a set of parameterised mutation operators for solving the dynamic environment problems produced by the Moving Peaks Benchmark (MPB) generator.

2 Proposed Method

In this study, we propose a selection hyper-heuristic incorporating a novel heuristic selection method, called the *Ant based Selection* (*AbS*), which is based on a simple ant colony optimization (ACO) algorithm [6]. Most of the mechanisms used in ACO are adapted within *AbS*. A distinct feature of *AbS* is that unlike ACO, *AbS* is based on a single point based search framework. In most of the selection hyper-heuristics, there is a *heuristic selection* step followed by an *acceptance* step. After the heuristic selection step using *AbS*, we employ the generic Improving-and-Equal acceptance scheme, which accepts a new solution of better or equal quality as compared to the previous solution.

Similar to the Choice Function and Reinforcement Learning heuristic selection schemes, *AbS* also incorporates an online learning mechanism using a matrix of utility values. In *AbS*, each heuristic pair is associated with a pheromone trail value (τ_{h_i,h_j}) which shows the desirability of selecting the j^{th} heuristic after the application of the i^{th} heuristic. All pheromone trails are initialized with a small value τ_0. *AbS* selects a random low-level heuristic at the first step. In the following steps, the most appropriate low-level heuristic is selected and applied to a solution in hand based on the pheromone trail information.

AbS consists of heuristic selection and pheromone update stages. For the first stage, we consider two variants of heuristic selection schemes. In both variants, the heuristic h_s with the highest pheromone trail ($h_s = \max_{i=1..k} \tau_{h_c,h_j}$) is selected with a probability of q_0 where h_c is the previously selected heuristic. Otherwise, methods inspired by two of the mate selection techniques most commonly used in Evolutionary Algorithms [8] are employed to determine the next heuristic to select. In the first variant, like in ACO, the next heuristic is determined based on probabilities proportional to the pheromone levels of each heuristic pair. This is similar to the roulette wheel mate selection in Evolutionary Algorithms. In this method, *AbSrw* selects the next heuristic h_s with a probability which is proportional to the pheromone trail value of τ_{h_c,h_s}. However, in the second variant (*AbSts*), the choice of the next heuristic is based on tournament selection. *AbSts* chooses the next heuristic h_s with the highest pheromone trail ($h_s = \max_{i=1..k} \tau_{h_c,h_j}$).

After selecting a heuristic, pheromone trails are modified. Firstly, pheromone values on the pheromone matrix are decreased by a constant factor (evaporation) for all pairs of heuristics as given in Equation 1.

$$\tau_{h_i,h_j} = (1-\rho)\tau_{h_i,h_j} \tag{1}$$

where $0 < \rho \leq 1$ is the pheromone evaporation rate.

After evaporation, the pheromone value between only h_c and h_s (τ_{h_c,h_s}) is increased using Equation 2.

$$\tau_{h_c,h_s} = \tau_{h_c,h_s} + \Delta\tau \tag{2}$$

where h_c is the previously selected heuristic and h_s is the last selected heuristic. $\Delta\tau$ is the amount of pheromone added and is defined as in Equation 3.

$$\Delta\tau = 1/f_c \tag{3}$$

where f_c is the fitness value of the new solution generated by applying the selected heuristic h_s.

3 Experimental Design

In this study, we perform experiments with our new hyper-heuristic for dynamic environments, combining the Ant-based selection scheme and the Improving-and-Equal acceptance technique. For comparison, we also experiment with previously used selection mechanisms which incorporate some form of online learning and are shown to be successful in dynamic environments [10], namely the Choice Function and Reinforcement Learning. In this paper, we also include an improved version of the Choice Function proposed in [7]. These selection mechanisms are also used together with the Improving-and-Equal acceptance technique.

In the experiments, we used the Moving Peaks Benchmark (MPB) generator [1] to generate the various dynamic environments. In MPB, the height, width and location of the peaks forming the multimodal and multidimensional landscape are changed every Δe iterations by adding a normally distributed random variable to the heights and the widths of the peaks and by shifting their locations with a vector of fixed length in a random direction. In some applications, a time-invariant basis function is also included in the generated landscapes. The *height_severity*, the *width_severity* and *vlength* parameters determine the severity of the changes in the heights, the widths and the locations of the peaks respectively. For the fixed parameters of the MPB in all the experiments, we used the settings taken from [1,2] as follows: cone peak function, 5 peaks, 5 dimensions, range of peak heights $\in [30, 70]$, range of peak widths $\in [0.8, 7.0]$ and range of values in each dimension $\in [0.0, 100.0]$. We did not use a basis function and allowed the peak shifts to be uncorrelated. For the Δe, *height_severity*, the *width_severity* and *vlength* settings we used the ones given in [10] and labeled as EXPSET2. Based on these settings, Δe is taken as 6006 fitness evaluations for low frequency (LF), 1001 for medium frequency (MF) and 126 for high frequency (HF); the *height_severity*, the *width_severity* and *vlength* parameters are taken as given in Table 1 which correspond to low severity (LS), medium severity (MS) and high severity (HS) changes.

Table 1. MPB parameter settings for high, medium and low severity changes

Setting	LS	MS	HS
vlength	1.0	5.0	10.0
height_severity	1.0	5.0	10.0
width_severity	0.1	0.5	1.0

A real-valued vector corresponds to the coordinates of a point in the search space generated by the MPB. The fitness of a candidate solution at a given time t is given by its *error*, which is calculated as its distance to the optimum in

terms of the objective function value at time t. Therefore, the goal turns into minimising the *error* values for a given problem.

The search algorithm moves through the landscape by perturbing a candidate solution at each step to obtain a new one using a parameterized Gaussian mutation, $N(0, \sigma^2)$, where σ denotes the standard deviation. We used the same settings for the mutation operators as in [10], which are implemented as seven different standard deviations; {0.5, 2, 7, 15, 20, 25, 30}. These mutation operators are used as the low-level heuristics in the hyper-heuristic framework.

The parameters of the proposed Ant-based selection scheme are chosen as follows: ρ is set to 0.1. Each entry in the pheromone matrix is initialized to $\tau_0 = 1/f_s$ where f_s is the fitness value of initial solution. For *AbSrw*, we experiment with seven q_0 values: $\{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$. For *AbSts*, we consider five tournament size values: $\{2, 3, 4, 5, 6\}$ as well as the above given seven q_0 values. We also included variants of the approaches in which $\Delta\tau$ is updated with a value changing at a slower rate during the pheromone evaporation process. In this case, $\Delta\tau$ is calculated as $\Delta\tau = 0.1 * (1/f_c)$. *AbSrw* and *AbSts* with the slower decrease rates are denoted as *sAbSrw* and *sAbSts*, respectively.

The parameter settings of the other heuristic selection methods are done using the same settings proposed in previous studies from the literature. In Reinforcement Learning, the scores of all heuristics are initialized to 15 with lower and upper bounds as 0 and 30 respectively as given in [13]. At each step, the score of a heuristic that improves performance is increased by 1 and otherwise it is decreased by 1. In Choice Function, α, β, and δ are initialized to 0.5 with updates of ± 0.01 at each iteration as given in [7]. In the Improved Choice Function, ϕ, which refers α and β, and δ are initialized to 0.5. If the heuristic improves performance, the values of ϕ are set to 0.99. Otherwise, the values of ϕ_t at time t are calculated as $\phi_t = \max\{\phi_{t-1} - 0.01, 0.01\}$. In addition, δ is calculated as $\delta_t = 1 - \phi_t$

We assume that all programs are made aware when a change in the environment occurs. For the Reinforcement Learning, Choice Function and the Improved Choice Function heuristic selection methods, when a change occurs, the current solution is re-evaluated. For the proposed Ant-based selection scheme, this is not required. The parameters of none of the heuristic selection methods are reset when the environment changes. Due to the nature of the acceptance mechanism, Improving-and-Equal, the first solution candidate generated after each environment change is accepted regardless of its solution quality.

For evaluating the performance of the approaches, we used the *offline error* metric [2]. The error of a candidate solution at a given time is calculated as its distance to the optimum in terms of the objective function value at that time. The offline error is calculated as a cumulative average of the errors of the best candidate solutions found so far since the last change. At the end of a run, a lower overall offline error value is desired indicating a *good* performance.

Each run is repeated 100 times for each setting where 20 changes occur, i.e. there are 21 consecutive stationary periods. This means that there are $maxIterations = (NoOfChanges + 1) * ChangePeriod$ number of iterations

per run. For analysis of statistical significance of the differences obtained between the results of various approaches, we performed ANOVA tests together with Tukey's HSD at a confidence level of 95%.

4 Results and Discussion

In this section we provide the results of the experiments and their discussions. Table 2 summarizes the results of the parameter tuning tests for q_0 in AbSrw and sAbSrw. In the table, $q_0 = 0.0$ means that the next heuristic is chosen using only the roulette-wheel selection. However, $q_0 = 1.0$ means that roulette wheel selection is not used and always the heuristic with the best score (pheromone value) is chosen to be applied. The results show that there are no statistically significant differences between most cases, however, the best values are provided by different q_0 values for different frequency-severity pairs. Therefore, to avoid overtuning, we decided to choose a setting which provided an acceptable performance in most of the cases for both approaches. For the rest of the experiments we continued with a setting of $q_0 = 0.5$ for both AbSrw and sAbSrw.

Table 2. Final offline error results of various q_0 settings for AbSrw and sAbSrw under the tested change frequency-severity pairs

Algorithm	q_0	LF			MF			HF		
		LS	MS	HS	LS	MS	HS	LS	MS	HS
AbSrw	0.0	3.577	7.664	9.875	4.647	8.860	12.321	9.886	17.031	24.041
	0.1	3.463	7.641	10.276	4.376	9.340	11.754	10.138	17.138	25.243
	0.3	3.929	8.026	10.244	4.608	8.774	12.354	9.286	16.453	24.612
	0.5	3.720	8.114	10.580	4.640	9.601	13.051	9.404	15.461	24.260
	0.7	4.186	8.434	10.562	4.975	10.838	13.178	10.434	17.419	25.493
	0.9	3.931	10.420	10.849	4.775	10.459	13.648	12.504	19.807	28.935
	1.0	3.956	10.368	12.332	5.533	10.849	13.997	15.898	23.064	31.920
sAbSrw	0.0	3.734	7.454	9.730	4.388	8.531	11.935	10.350	17.105	25.586
	0.1	3.723	7.090	9.882	4.490	8.510	12.330	9.948	17.183	24.368
	0.3	3.978	7.598	10.189	4.394	8.245	12.467	8.875	15.444	24.027
	0.5	3.644	7.402	10.990	4.180	8.698	12.421	7.932	14.935	22.904
	0.7	3.879	8.578	10.945	4.483	9.490	12.531	8.914	16.004	24.131
	0.9	4.278	8.827	11.936	4.399	10.095	13.334	8.691	15.432	23.713
	1.0	4.878	9.959	12.563	6.528	12.620	14.802	13.535	19.350	25.191

Then we performed parameter tuning experiments to set q_0 and tournament size values for the AbSts and sAbSts variants. Our experiments revealed that for this hyper-heuristic variant, the best settings highly depend on the dynamics of the environment, i.e. the change frequency and severity values. To be able to choose a value which would give a good performance across different types of

chnages, we calculated the average and standard deviation values for the final offline error results for each change frequency-severity setting over all q_0 and tournament size combinations, i.e. over 35 combinations each for AbSts and sAbSts. Table 3 lists these results.

Table 3. Average and standard deviation of final offline error values over all q_0 and tournament size combinations under all tests frequency-severity pairs

		AbSts	*sAbSts*
LF	LS	4.061 ∓ 0.238	4.121 ∓ 0.525
	MS	8.911 ∓ 0.510	8.997 ∓ 0.644
	HS	11.291 ∓ 0.505	11.577 ∓ 0.795
MF	LS	5.361 ∓ 0.343	5.106 ∓ 0.736
	MS	10.514 ∓ 0.649	10.530 ∓ 0.867
	HS	13.345 ∓ 0.536	13.318 ∓ 0.717
HF	LS	14.720 ∓ 2.050	11.051 ∓ 3.083
	MS	20.542 ∓ 1.408	17.474 ∓ 2.134
	HS	27.701 ∓ 1.839	24.435 ∓ 1.492

Considering that the best setting for the q_0 and tournament size combination is sensitive to the change dynamics, we plan to develop an adaptive mechanism as future work. In this study, we chose a simpler approach. For those cases where tournament selection is to be applied, each time we let the tournament size be determined randomly with equal probability from among the five pre-determined tournament size levels. We performed the q_0 analysis for AbSts and sAbSts based on this scheme. Table 4 shows the final offline error results for various q_0 settings for AbSts and sAbSts when the tournament sizes are determined randomly. From the table, we selected $q_0 = 0.1$ for AbSts and $q_0 = 0.9$ for sAbSts. For the rest of the experiments, these settings are used. Comparing the final offline error values for these settings with the average and standard deviation values given in Table 3, we observe that the chosen settings provide either better results than these or the results fall within the intervals defined by these.

Finally, we compare our approaches with those obtained using the heuristic selection methods taken from literature, namely RL, CF and ICF. Table 5 shows the results of these comparisons. The better results are marked in bold in the table. As can be seen, RL and ICF are worse than the others for all cases and these differences are statistically significant, with ICF being the worst of all.

ICF aims to emphasize the intensification component of the generic CF by automatically increasing the weight of relevant components as soon as there is improvement. Diversification, on the other hand, is introduced at a gradually increasing rate. This property works in solving stationary combinatorial optimisation problems as shown in [7], but not in dynamic environments. The changes in the environment mislead ICF and it gets worse than CF in all cases. It was shown in [9,10] that CF outperforms RL for all tested change dynamics using

Table 4. Final offline error results of various q_0 settings for AbSts and sAbSts with random tournament sizes under the tested change frequency-severity pairs

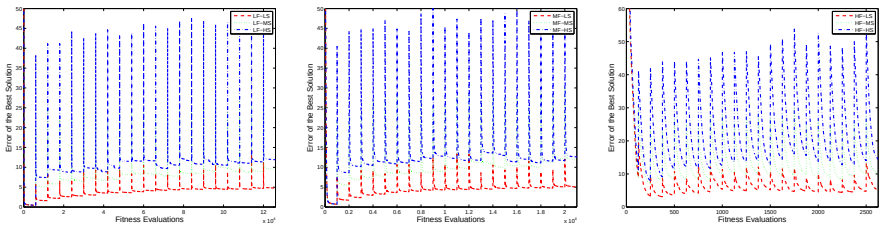
Algorithm	q_0	LF			MF			HF		
		LS	MS	HS	LS	MS	HS	LS	MS	HS
AbSts	0.0	3.844	8.516	10.901	5.236	10.231	13.033	13.434	18.360	25.459
	0.1	4.063	8.019	10.940	5.069	9.566	12.853	13.486	19.035	25.256
	0.3	3.744	8.287	11.107	5.172	9.974	13.192	13.120	19.325	25.327
	0.5	3.932	8.086	11.877	5.012	10.201	13.360	12.843	18.345	25.949
	0.7	3.916	8.322	11.777	5.150	9.689	13.176	12.872	18.833	26.593
	0.9	4.030	8.778	12.011	4.924	11.963	12.958	13.861	21.461	30.139
	1.0	3.987	9.977	11.735	5.460	10.791	14.618	14.640	22.261	30.589
sAbSts	0.0	4.150	8.194	10.369	5.051	10.111	12.995	12.236	17.250	24.222
	0.1	4.235	8.487	10.715	5.215	10.104	13.045	12.804	18.561	24.716
	0.3	3.675	8.367	11.948	4.923	10.454	12.438	11.381	17.892	24.151
	0.5	3.906	8.425	11.009	4.660	9.836	13.037	10.098	16.743	24.883
	0.7	4.077	8.632	11.509	4.643	10.308	13.190	9.678	16.929	23.911
	0.9	4.030	8.996	11.735	4.635	10.025	12.387	9.117	16.760	23.416
	1.0	4.746	10.567	12.595	7.699	12.250	14.559	13.404	18.133	24.266

the MPB. Therefore, the poor performance of RL in the current experiments is also to be expected.

The results show that all versions of the proposed heuristic selection scheme provide better results than CF, except for the LF-LS and MF-LS cases, however, the results are close. Among the versions of the proposed heuristic selection scheme, sAbSrw provides the better results in most cases.

Table 5. Final offline error results for the proposed heuristic selection schemes and RL, CF and ICF. Here, for both AbSrw and sAbSrw $q_0 = 0.5$, for AbSts $q_0 = 0.1$ and for sAbSts $q_0 = 0.9$ with random tournament size settings

Algorithm	LF			MF			HF		
	LS	MS	HS	LS	MS	HS	LS	MS	HS
AbSrw	3.720	8.114	**10.580**	4.640	9.601	13.051	9.404	15.461	24.260
sAbSrw	3.644	**7.402**	10.990	4.180	**8.698**	12.421	**7.932**	**14.935**	**22.904**
AbSts	4.063	8.019	10.940	5.069	9.566	12.853	13.486	19.035	25.256
sAbSts	4.030	8.996	11.735	4.635	10.025	**12.387**	9.117	16.760	23.416
CF	**3.518**	9.801	11.881	**4.094**	10.948	13.601	7.976	15.670	24.390
ICF	10.065	15.290	18.141	11.043	19.688	20.762	19.119	27.285	32.785
RL	4.427	9.046	12.179	5.968	12.440	14.540	10.285	15.685	24.569



(a) (b) (c)

Fig. 1. Illustration of the tracking ability of sAbSrw for (a) low, (b) medium, and (c) high frequency of change

Due to lack of space, we cannot provide all the statistical comparison tables. Table 6 provides a summary which shows that sAbSrw performs the best with a high level of tracking ability (Figure 4) in the overall based on the number of cases in which this performance difference is either statistically significant or better than the other approaches. sAbSrw is better than CF in many cases, but in several others, their performance is comparable. However, the most important issue is the fact that sAbSrw (and also all the other proposed variants) are more suitable to be used in dynamic environments than RL, CF and ICF because the proposed heuristic selection schemes do not require any special actions to be performed when the environment changes, whereas for the others, right after an environment change, the last solution candidate in the previous environment needs to be re-evaluated. This is a drawback for two reasons: it makes change detection necessary and it also wastes fitness evaluations, especially in environments where the change frequencies are very high.

Table 6. Summary of statistical significance comparisons, where $s+$ ($\geq$) is the total number of times for which an algorithm was statistically significantly (slightly) better than the others (with no statistical significance), $s-$ ($\leq$) is the vice versa.

Algorithm	$s+$	$s-$	$\geq$	$\leq$	Algorithm	$s+$	$s-$	$\geq$	$\leq$
sAbSrw	15	0	34	5	CF	13	1	18	22
AbSrw	12	0	24	18	ICF	0	54	0	0
AbSts	10	9	16	19	RL	11	7	3	33
sAbSts	10	0	23	21					

5 Conclusion

In this paper, we proposed a new heuristic selection scheme for selection hyper-heuristics, especially for use in dynamic environments. In previous studies [9,10,15],

existing heuristic selection mechanisms were tested in various types of dynamic environments and those that incorporated some form of online learning were shown to be successful. One drawback of these approaches for dynamic environments is that they require the re-evaluation of the last candidate solution in the previous environment for score calculation. As well as wasting computing resources for the re-evaluation, this also means that the algorithm needs to detect when the environment changes. The proposed heuristic selection does not require any special actions when the environment changes. On top of this advantage, the test results also show that the proposed heuristic selection scheme provides slightly better results than the best of the tested heuristic selection schemes found previously to be successful in dynamic environments. The results are very promising and they promote further study. A drawback of the proposed variants of the method is that a couple of parameters is introduced and in some cases, performance is sensitive to their setting. Our future work will focus on enhancing the proposed approach by developing adaptive mechanisms to alleviate the need for parameter tuning and better acceptance schemes for dynamic environment problems.

References

1. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Congress on Evolutionary Computation CEC 99. vol. 3, pp. 1875–1882. IEEE (1999)
2. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer (2002)
3. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Rong, Q.: A survey of hyper-heuristics. Tech. rep. (2009)
4. Cowling, P., Kendall, G., Soubeiga, E.: A hyper-heuristic approach to scheduling a sales summit. In: Practice and Theory of Automated Timetabling III : Third International Conference, PATAT 2000. LNCS, vol. 2079. Springer (2000)
5. Cruz, C., Gonzalez, J., Pelta, D.: Optimization in dynamic environments: a survey on problems, methods and measures. Soft Computing - A Fusion of Foundations, Methodologies and Applications 15, 1427–1448 (2011)
6. Dorigo, M., Stützle, T.: Ant Colony Optimizations. MIT Press (2004)
7. Drake, J.H., Özcan, E., Burke, E.K.: An improved choice function heuristic selection for cross domain heuristic search. In: Parallel Problem Solving from Nature - PPSN XII. Springer Berlin Heidelberg (2012)
8. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Spring (2003)
9. Kiraz, B., Uyar, A.S., Özcan, E.: An investigation of selection hyper-heuristics in dynamic environments. In: Proc. of the int. conf. on Applications of EC - Vol. I. EvoApplications'11, vol. 6624, pp. 314–323 (2011)
10. Kiraz, B., Uyar, A.S., Özcan, E.: Selection hyper-heuristics in dynamic environments. Journal of the Operational Research Society (to appear)
11. Morrison, R.W.: Designing evolutionary algorithms for dynamic environments. Springer (2004)
12. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: Metaheuristics: Computer Decision-Making. pp. 523–544. Kluwer Academic Publishers (2001)

13. Özcan, E., Misir, M., Ochoa, G., Burke, E.K.: A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing 1(1), 39–59 (2010)
14. Özcan, E., Ş. Uyar, A., Burke, E.: A greedy hyper-heuristic in dynamic environments. In: GECCO 2009 Workshop on Automated Heuristic Design: Crossing the Chasm for Search Methods. pp. 2201–2204 (2009)
15. Uludag, G., Kiraz, B., Etaner-Uyar, A.S., Özcan, E.: A framework to hybridize pbil and a hyper-heuristic for dynamic environments. In: PPSN (2). LNCS, vol. 7492, pp. 358–367. Springer-Verlag, Berlin, Heidelberg (2012)
16. Yang, S., Ong, Y.S., Jin, Y. (eds.): Evolutionary Computation in Dynamic and Uncertain Environments, Studies in Computational Int., vol. 51. Springer (2007)