A Framework to Hybridize PBIL and a Hyper-heuristic for Dynamic Environments

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

[1] Istanbul Technical University, Turkey
{uludagg,etaner}@itu.edu.tr, berna.kiraz@marmara.edu.tr
[2] University of Nottingham, UK
Ender.Ozcan@nottingham.ac.uk

Abstract. Selection hyper-heuristic methodologies explore the space of heuristics which in turn explore the space of candidate solutions for solving hard computational problems. This study investigates the performance of approaches based on a framework that hybridizes selection hyper-heuristics and population based incremental learning (PBIL), mixing offline and online learning mechanisms for solving dynamic environment problems. The experimental results over well known benchmark instances show that the approach is generalized enough to provide a good average performance over different types of dynamic environments.

Keywords: hyper-heuristics, dynamic environments, multiple populations, incremental learning.

1 Introduction

Many real world optimization problems are dynamic in nature. When solving a problem in such environments, it is better to take the dynamism into account and choose an appropriate optimisation approach which is able to adapt and track the moving optima. Different types of changes may occur in the environment over time. The dynamism in the environment can be classified based on its severity, frequency, predictability, cycle length and cycle accuracy [2]. There are many techniques proposed in literature to solve dynamic optimization problems. A recent survey can be found in [5].

Recently, there has been a growing interest in Estimation of Distribution Algorithms (EDAs). The performance improvement of EDAs via the development of different algorithmic frameworks, such as multi-population approaches, inclusion of mechanisms addressing issues, such as hyper-mutation to deal with diversity loss and other mechanisms are of interest for many researchers and practitioners to solve dynamic environment problems [1,6,14,16,13,18].

There is an emerging field of research in the semi-automated design of search methodologies: *hyper-heuristics*. Burke et al. [3] defined hyper-heuristics as methodologies that search the space of heuristics by *selecting* or *generating* them

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 358–367, 2012.

to solve difficult problems. The focus of this study is selection hyper-heuristics which attempt to improve an initially generated candidate solution iteratively through *heuristic selection* and *move acceptance* stages [4,10]. In this paper, we will use *hyper-heuristics* to denote *selection hyper-heuristics*. Özcan et al. [11] proposed a hyper-heuristic framework for dynamic environments for the first time, to the best of the authors' knowledge. Empirical evidence suggests that hyper-heuristics are effective solvers in dynamic environments for real valued optimisation [7] as well as combinatorial optimisation [8].

Although variants of EDAs have been proposed to solve dynamic environment problems, it has been observed that there is almost no single approach that performs consistently well across different types of dynamic environments. This is mostly because different types of methods are capable of handling particular types of changes relatively better than others in such environments.

In this study, inspired from previous studies, we investigate the performance of a general framework which is based on a bi-population approach hybridizing a variant of EDA, in particular PBIL, and a selection hyper-heuristic across some well known benchmark functions. The goal of the study is to enhance the performance of PBIL enabling this approach to handle any given type of change dynamic and hence, raise its level of generality. The framework can combine any EDA based approach with any type of selection hyper-heuristic. We utilize an offline learning mechanism to detect the useful operators (or operator components) for different environments and then use an online learning selection hyper-heuristic to select the best operator at a given time during the search process while solving an unseen instance. The following sections discuss the details of the proposed framework.

2 Proposed Framework

In this study, we propose a new framework exploiting the advantages of hyper-heuristics and multi-population approaches. The framework can combine any multi-population EDA with selection hyper-heuristics. Here we propose *hyper-heuristic based multi-population PBIL* (HH-PBIL2), which is based on SPBIL2 introduced in [18]. SPBIL2 is a bi-population standard PBIL (SPBIL) algorithm.

Kiraz et al. [7], show that heuristic selection methods with learning, namely choice function and reinforcement learning (see [10] for details) outperform others. Both incorporate some form of a scoring mechanism. In choice function, when scoring a heuristic, the difference between the fitness values of the offspring and the current candidate solution is taken into account. In a dynamic environment setting, this means that whenever a change occurs, the current candidate solution has to be re-evaluated in the new environment. For the proposed approach this involves re-evaluating all the candidate solutions in the current population, which is computationally ineffective. Therefore, we do not use choice function as a heuristic selection method. In reinforcement learning (RL) [9] heuristic selection method, each low-level heuristic has a utility score. The scores of each heuristic are initialized to the same value and updated during the search process based on its performance.

At each step, the low-level heuristic with the maximum score is selected. If the selected heuristic produces a better solution than the previous one, it is rewarded by increasing its score, otherwise it is penalized by decreasing it. The scores are restricted to vary between predetermined lower and upper bounds.

In SPBIL, a posterior probability distribution model of promising solutions is built using statistical information obtained from the population of solution candidates. This is termed as a *probability vector*, which is used to create a population of solutions through sampling at each iteration. In SPBIL2, the population is divided into two sub-populations. Each sub-population is sampled from its own probability vector. The two probability vectors are evolved in parallel for a given maximum number of generations. As in SPBIL, the first probability vector $\overrightarrow{P}_1$ is initialized with the central probability vector, and the second probability vector $\overrightarrow{P}_2$ is initialized randomly. The size of the initial sub-populations are equal. After all candidate solutions are evaluated, sub-population sample sizes are slightly adjusted within the range of $[0.3 * n,\ 0.7 * n]$. Then, each probability vector is learnt towards the best solution(s) in the relevant sub-population. Similar to SPBIL, mutation is applied to both probability vectors before sampling. Details of SPBIL2 can be found in [18,17].

The approach proposed in this paper (HH-PBIL2) consists of two phases. In the first phase, probability vectors corresponding to a set of different environments are learned offline, using SPBIL. Then, those learned probability vectors are stored for later use. In the second phase, the probability vectors serve as low-level heuristics for the RL based hyper-heuristic.

HH-PBIL2 is proposed to enhance the performance of SPBIL2 in dynamic environments. As in SPBIL2, the population is divided into two sub-populations and two probability vectors are used in parallel. The first probability vector $\overrightarrow{P}_1$ is again initialized with the central probability vector, but the second probability vector $\overrightarrow{P}_2$ is selected randomly from the previously stored probability vectors. Each sub-population is sampled independently using the relevant probability vector. The first probability vector $\overrightarrow{P}_1$ is learned towards the best solution candidate(s) in the first population. There is no online learning step for the second probability vector $\overrightarrow{P}_2$. At each iteration, RL heuristic selection mechanism selects the probability vector with the largest score from among the previously stored probability vectors and this probability vector is assigned as $\overrightarrow{P}_2$. The score update scheme for the RL heuristic selection method is explained above.

We used two variants of HH-PBIL2 which differ in the information used to update a low-level heuristic's score. In the first variant, RL-PF, the best performing candidate solution(s) from the two populations combined, are used to update the score. In the second variant, RL-P2, the best performing solution candidate(s) from only the second population is used to update the score.

Similar to SPBIL2, after the candidate solutions are evaluated, the next population sizes are slightly adjusted. However, mutation is applied only to $\overrightarrow{P}_1$. Then, two sub-populations are sampled based on the relevant probability vectors. The approach repeats the cycle until some termination criteria are met. The pseudocode of HH-PBIL2 is shown in Algorithm 1.

Algorithm 1. Pseudocode of the proposed approach HH-PBIL2.

```
t := 0
initialize P⃗1(0) := 0.5⃗
P⃗2(0) is selected from RL randomly
S1(0) := sample(P⃗1(0)) and S2(0) := sample(P⃗2(0))
while (termination criteria not fulfilled) do
    evaluate S1(t) and evaluate S2(t)
    adjust next population sizes for P⃗1(t) and P⃗2(t) respectively
    place k best samples from S1(t) and S2(t) into B⃗(t)
    if (RL-PF) then
        send the best fitness value from the whole population to RL
    end if
    if (RL-P2) then
        send the best fitness value from the second population to RL
    end if
    learn P⃗1(t) toward B⃗(t)
    mutate P⃗1(t)
    P⃗2(t) is selected with maximum score from RL
    S1(t) := sample(P⃗1(t)) and S2(t) := sample(P⃗2(t))
    t := t + 1
end while
```

3 Experimental Design

In the experiments, the proposed approaches RL-PF and RL-P2 are compared with SPBIL and SPBIL2. The original source codes which we compared are taken from Yang's web site[1]. Our approaches are implemented based on SPBIL2. These two techniques are briefly explained in section 2.

All approaches are applied to three Decomposable Unitation-Based Functions (DUFs). All DUFs are composed of 25 copies of 4-bit building blocks. Each building block is denoted as a unitation-based function $u(x)$ which gives the number of ones in the corresponding building block. Its maximum value is 4. The fitness of a bit string is calculated as the sum of the $u(x)$ values of the building blocks. The optimum fitness value for all DUFs is 100. The DUFs can be formulated as follows [13].

$$f_{DUF1} = u(x) \qquad f_{DUF2} = \begin{cases} 4 \text{ , if } u(x) = 4 \\ 2 \text{ , if } u(x) = 3 \\ 0 \text{ , if } u(x) < 3 \end{cases} \qquad f_{DUF3} = \begin{cases} 4 & \text{, if } u(x) = 4 \\ 3 - u(x) & \text{, if } u(x) < 4 \end{cases}$$

$DUF1$ is the *OneMax* problem whose objective is to maximize the number of ones in a bit string. $DUF2$ has a unique optimal solution surrounded by four local optima and a wide plateau with eleven points having a fitness of zero. $DUF2$ is more difficult than $DUF1$. $DUF3$ is fully deceptive [18].

The XOR dynamic problem generator [15,17] is applied to the three DUFs to obtain dynamic test problems. The XOR generator can create a dynamic environment problem with varying degrees of difficulty from any binary-encoded stationary problem using a bitwise exclusive-or (XOR) operator. Given a function $f(\boldsymbol{x})$ in a stationary environment and $\boldsymbol{x} \in \{0,1\}^l$, the fitness value of the $\boldsymbol{x}$ at a given generation g is calculated as $f(\boldsymbol{x}, g) = f(\boldsymbol{x} \oplus m_k)$, where m_k is a binary mask for k^{th} stationary environment and $\oplus$ is the XOR operator. Firstly,

[1] `http://www.brunel.ac.uk/~csstssy/publications.html`

the mask m is initialized with a zero vector. Then, every τ generations, the mask m_k is changed as $m_k = m_{k-1} \oplus t_k$, where t_k is a binary template.

SPBIL, SPBIL2 and HH-PBIL2 share some common settings which are used as suggested in [18]. The problem consists of 25 building blocks, therefore solution candidates are of length 100. Mutation rate is 0.02 and mutation shift is 0.05. The learning rate α is taken as 0.25 and 3 best candidate solutions are used in the online learning of probability vectors. The population size for SPBIL is set to 100. For SPBIL2 and HH-PBIL2, each sub-population size is initialized as 50 and they are allowed to vary between 30 and 70. In RL, score of each heuristic is initialized to 15 and allowed to vary between 0 and 30. If the selected heuristic yields a solution with an improved fitness, its score is increased by 1, otherwise decreased by 1. The RL settings are taken as recommended in [12].

In the first phase of HH-PBIL2, probability vectors corresponding to a set of different environments are learned offline using SPBIL. To generate different environments using the XOR generator, a set of M XOR masks are randomly generated. Then, for each mask (i.e. environment), SPBIL is executed for 100 independent runs where each run consists of $10,000$ generations. During offline learning, each environment is stationary. At the end, for each environment, the probability vector producing the best solution found so far over all runs is stored. These vectors are used in in all the rest of the experiments.

This study considers the frequency of changes τ, severity of changes ρ and cycle length CL as the type of changes in the environment. In the cyclic environments, we assume that environments return to their previous locations exactly. None of the tested methods require that the time of a change is known.

As a result of some preliminary experiments, we determined the change periods as 50 generations for low frequency (LF), 25 generations for medium frequency (MF) and 5 generations for high frequency (HF) for DUF1 and DUF2. The change periods for DUF3 are determined as 100 generations for LF, 35 generations for MF and 10 generations for HF. In convergence plots, these settings for LF, MF and HF correspond respectively to stages where the algorithm has been converged for some time, where it has not yet fully converged and where it is very early on in the search. In addition, the severity of changes are chosen as 0.1 for low severity (LS), 0.2 for medium severity (MS), 0.5 for high severity (HS), and 0.75 for very high severity (VHS) for random dynamic environments. These are determined based on the definition of the XOR generator. For cyclic dynamic environments, the cycle lengths CL are selected as 2, 4 and 8. To construct cyclic environments, the masks representing the environments are selected among the randomly generated M masks used in the offline learning phase of HH-PBIL2. For each run of the algorithms, 128 changes occur after the initial environment. Therefore, the maximum number of generations is calculated as $maxGenerations = changeFrequency * changeCount$. We performed experiments to explore the effects of the severity and the frequency of the changes on the performance of the approaches for randomly changing environments, and the effects of the cycle length and the frequency of the changes on the performance of the approaches for cyclic environments.

In order to compare the performance of the algorithms, the results are reported in terms of the offline error [2], calculated as the cumulative average of the errors of the best candidate solutions found so far. The error of a candidate solution is calculated as the difference of its fitness value from the fitness value of the optimum solution at each time step. Fitness values are calculated using the corresponding DUF definitions given above. In all our experiments, while the location of the global optimum may change, its fitness value remains the same and is 100 for all time steps. An algorithm solving a dynamic environment problem aims to achieve the least overall offline error value obtained at the end of a run. All reported results are averages over 100 independent runs. Anova and Tukey's HSD tests are applied to the results at a significance level of 95% to test for statistically significant differences.

4 Results and Discussion

All test results are summarized in Table 2 for randomly changing environments and in Table 3 for cyclic environments on all DUFs. The values in the tables show the offline errors achieved at the end of a run, averaged over 100 runs. Due to lack of space, the statistical significance comparison tables are not given in the paper[2].

Firstly, we analyze the effects of the learned probability vector counts (M) on HH-PBIL2 in both randomly changing and cyclic environments. We experimented with M values of $8, 16, 32, 64$. The results of the ANOVA and Tukey's HSD tests for statistical significance at a 95% confidence level are reported in Table 1. In the table, each entry shows the total number of times the approach achieves the corresponding significance state ($s+$, $s-$, $\geq$ and $\leq$) over the others on the three DUFs for different change severity and frequency settings in randomly changing environments and for different cycle length and change frequency settings in cyclic environments. Here, the following notation is used: Given A vs B, $s+$ ($s-$) denote that A (B) is performing statistically better than B (A), while $A \geq B$ ($A \leq B$) indicates that A (B) performs slightly better than B (A) and this performance difference is not statistically significant. From the table, we can see that $M = 8$ is better overall for the tested environments under all change settings. Therefore, in the tables 2 and 3, we only report the results for $M = 8$. The statistical significance tests show that the number of learned probability vectors does not significantly affect the performance of HH-PBIL2 variants for all change frequency-severity settings. However, for cyclic environments smaller M values give better offline error values.

Secondly, we analyze the performance of HH-PBIL2 in dynamic environments showing different change properties. Both for the randomly changing environments and the cyclic environments in all DUFs, SPBIL2 is significantly better than SPBIL, except for HF cyclic changes in DUF1 and DUF2 where the performance difference is not statistically significant. In the cyclic environments,

[2] Statistical significance comparison tables can be download from `http://web.itu.edu.tr/etaner/ppsn2012_analysis.zip`

RL-P2 is significantly better than RL-PF on average, SPBIL and SPBIL2 for all M values and change frequencies. In the randomly changing environments, for all HS and VHS severity settings RL-P2 is significantly better than RL-PF in all the DUFs. For the LS and MS severity settings, their performance differences are not statistically significant. However, statistically significant performance differences appear in favor of RL-P2 for these severity settings in MF and HF settings in all DUFs. In the randomly changing environments for all change severities (LS, MS, HS, VHS) at the HF frequency setting, RL-P2 is better than SPBIL2 and this performance difference is statistically significant. The same result is also observed for all change frequencies (LF, MF, HF) at the HS and VHS severity settings. For change frequency-severity combinations of LF and MF with LS and MS, SPBIL2 is significantly better than RL-P2.

To illustrate the tracking behavior of the approaches, in Figure 1 and Figure 2, sample plots for the error values of the generation best solution candidates versus the number of generations, for four consecutive environments after the third change on DUF2 are given. The plots show that for randomly changing environments, increased change severities result in significant differences between the algorithms in favor of HH-PBIL2 variants. Increased cycle lengths in cyclic environments have a similar effect on the algorithms. But as the frequency increases, the differences get smaller. Increased change frequencies have a similar effect on all approaches in the cyclic environments too.

Table 1. Overall ($s+$, $s-$, $\geq$ and $\leq$) counts for $M = 8, 16, 32, 64$ in RL-PF and RL-P2

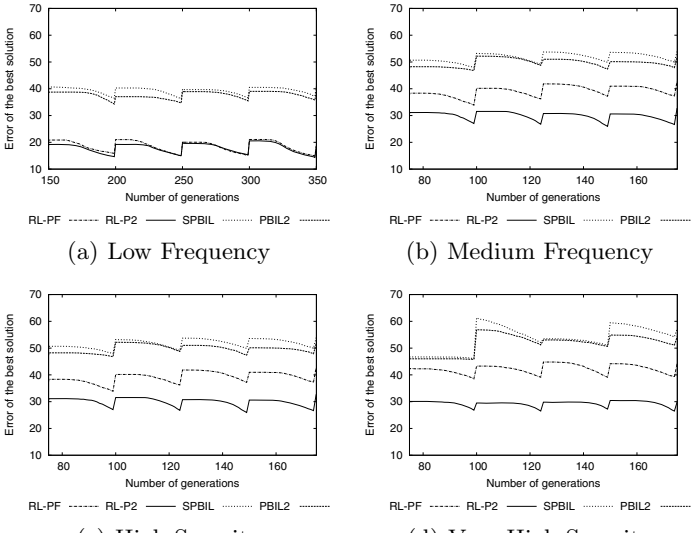
	RL-P2-8	RL-P2-16	RL-P2-32	RL-P2-64	RL-PF-8	RL-PF-16	RL-PF-32	RL-PF-64
$s+$	376	356	342	283	202	167	137	104
$s-$	64	73	88	139	218	236	273	320
$\geq$	44	81	80	70	70	90	76	71
$\leq$	83	57	57	75	77	74	81	72

Table 2. Offline errors averaged over 100 runs, on the three DUFs for different change severity and frequency settings in randomly changing environments

Alg.	LF				MF				HF			
	LS	MS	HS	VHS	LS	MS	HS	VHS	LS	MS	HS	VHS
DUF1												
RL-PF	4.22	8.12	9.02	9.22	9.91	16.74	20.58	22.06	27.91	32.13	36.65	38.73
RL-P2	4.24	8.15	**7.23**	**4.25**	9.95	16.73	**14.39**	**12.55**	**26.95**	**29.67**	**33.17**	**35.12**
SBIL	4.11	7.91	16.72	21.76	9.55	16.08	26.73	30.45	28.00	33.84	38.05	38.73
SPBIL2	**3.46**	**7.21**	16.21	20.72	**9.05**	**15.81**	26.00	29.18	27.75	33.35	37.23	38.12
DUF2												
RL-PF	9.28	19.26	19.14	15.05	21.23	34.53	39.88	42.37	50.03	56.56	63.43	64.61
RL-P2	9.28	19.16	**18.52**	**13.39**	21.40	34.37	**30.48**	**29.00**	**49.04**	**53.19**	**58.64**	**60.67**
SPBIL	9.00	18.42	38.86	45.88	20.43	34.48	51.51	54.83	52.30	60.45	65.21	65.72
SPBIL2	**7.63**	**17.21**	37.06	43.15	**19.53**	**33.71**	49.71	52.59	51.79	59.35	64.07	64.57
DUF3												
RL-PF	25.57	25.92	18.95	**17.79**	30.44	32.22	29.41	27.24	39.77	41.90	44.42	42.90
RL-P2	25.55	25.89	**19.02**	17.83	30.41	**32.00**	**25.03**	**25.30**	**38.94**	**39.78**	**41.78**	**40.00**
SPBIL	25.46	25.81	23.98	19.46	30.12	33.17	35.29	31.53	40.18	44.51	47.18	45.94
SPBIL2	**25.00**	**25.26**	23.19	18.52	**29.44**	32.38	34.36	30.71	39.48	43.65	46.35	45.09

Table 3. Offline errors averaged over 100 runs, on the three DUFs for different cycle length and change frequency settings in cyclic environments

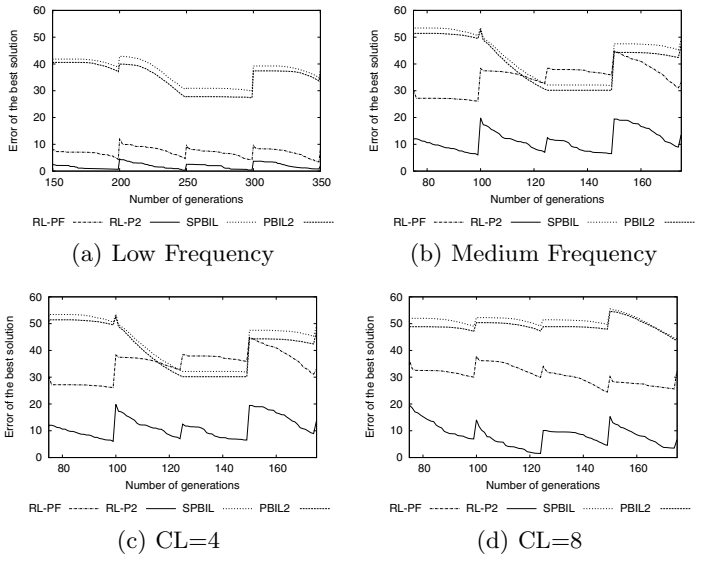
Alg.	LF			MF			HF		
	CL=2	CL=4	CL=8	CL=2	CL=4	CL=8	CL=2	CL=4	CL=8
DUF1									
RL-PF	3.50	4.02	3.84	15.17	17.66	13.92	16.71	19.37	27.99
RL-P2	**0.17**	**0.17**	**0.17**	**1.80**	**2.13**	**1.93**	**8.95**	**14.76**	**19.33**
SPBIL	10.19	16.51	15.84	13.20	22.43	24.13	15.79	26.23	28.42
SPBIL2	9.08	15.73	15.29	10.73	21.05	23.08	16.24	26.20	28.19
DUF2									
RL-PF	2.18	2.06	2.58	14.38	22.72	19.81	27.29	36.27	47.93
RL-P2	**0.27**	**0.29**	**0.27**	**2.85**	**3.40**	**3.40**	**15.74**	**27.59**	**32.95**
SPBIL	20.67	36.15	36.73	24.29	43.07	46.89	27.69	45.83	51.23
SPBIL2	17.79	33.71	34.63	20.91	40.40	44.58	28.60	45.82	50.77
DUF3									
RL-PF	10.94	11.93	11.91	18.65	26.95	20.39	24.16	34.31	36.35
RL-P2	**10.53**	**11.58**	**11.57**	**12.99**	**14.35**	**14.21**	**17.51**	**29.35**	**27.79**
SPBIL	25.72	24.25	23.88	31.52	34.77	34.86	28.60	37.24	42.66
SPBIL2	25.00	23.48	23.07	30.35	33.49	33.95	28.44	36.38	41.49



(a) Low Frequency (b) Medium Frequency

(c) High Severity (d) Very High Severity

Fig. 1. Sample plots for the error values of the generation best solution candidates versus the number of generations for randomly changing environments based on fixed severity - HS- (first row) and based on fixed frequency -MF- (second row) settings.



(a) Low Frequency (b) Medium Frequency

(c) CL=4 (d) CL=8

Fig. 2. Sample plots for the error values of the generation best solution candidates versus the number of generations for cyclic changing environments based on fixed CL=4 (first row) and based on fixed frequency -MF- (second row) settings.

5 Conclusion

In this study, we investigated the performance of a bi-population framework that hybridizes a variant of population based incremental learning with a selection hyper-heuristic. The framework can combine any EDA based technique with any heuristic selection mechanism. In this study, a standard PBIL is hybridized with a reinforcement learning selection hyper-heuristic. To explore the generality of the proposed approach we performed experiments across environments exhibiting a range of different change dynamics on some well known benchmark functions for two generic approaches and our proposed approach. Previous studies indicate that stand-alone generic approaches are not sufficient to deal with different change dynamics. The results of the experiments in this study confirm this and show that the proposed approach exhibits good performance in all the tested change scenarios. This makes the proposed approach a solver which is generalized enough to provide a good average performance over different types of dynamic environments. As future work, we will experiment with hybridizing other types of EDA based methods and heuristic selection mechanisms, as well as incorporate our approach into memory based techniques. The results are promising which promote further study.

Acknowledgements. B. Kiraz is supported by TÜBİTAK 2211-National Scholarship Program for PhD students. The study is supported in part by EPSRC, grant EP/F033214/1 (The LANCS Initiative Postdoctoral Training Scheme).

References

1. Barlow, G.J., Smith, S.F.: Using memory models to improve adaptive efficiency in dynamic problems. In: IEEE Symposium on Computational Intelligence in Scheduling, CISCHED, pp. 7–14 (2009)
2. Branke, J.: Evolutionary optimization in dynamic environments. Kluwer (2002)
3. Burke, E.K., Gendreau, M., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: A survey of the state of the art. To appear in the Journal of the Operational Research Society (2012)
4. 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)
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. Fernandes, C.M., Lima, C., Rosa, A.C.: Umdas for dynamic optimization problems. In: Proc. of the 10th Conference on Genetic and Evolutionary Computation, GECCO 2008, pp. 399–406. ACM (2008)
7. 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)
8. Kiraz, B., Topcuoglu, H.R.: Hyper-heuristic approaches for the dynamic generalized assignment problem. In: 2010 10th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 1487–1492 (2010)
9. Nareyek, A.: Metaheuristics, pp. 523–544. Kluwer (2004)
10. Özcan, E., Bilgin, B., Korkmaz, E.E.: A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12, 3–23 (2008)
11. Özcan, E., Uyar, Ş., 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)
12. 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)
13. Peng, X., Gao, X., Yang, S.: Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments. Soft Comput. 15, 311–326 (2011)
14. Wu, Y., Wang, Y., Liu, X., Ye, J.: Multi-population and diffusion umda for dynamic multimodal problems. Journal of Systems Engineering and Electronics 21(5), 777–783 (2010)
15. Yang, S.: Constructing dynamic test environments for genetic algorithms based on problem difficulty. In: Proc. of the 2004 Congress on Evolutionary Computation, pp. 1262–1269 (2004)
16. Yang, S., Richter, H.: Hyper-learning for population-based incremental learning in dynamic environments. In: Proc. 2009 Congr. Evol. Comput., pp. 682–689 (2009)
17. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)
18. Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. on Evolutionary Comp. 12, 542–561 (2008)