Direct Memory Schemes for Population-based Incremental Learning in Cyclically Changing Environments

Michalis Mavrovouniotis and Shengxiang Yang

Centre for Computational Intelligence (CCI)
School of Computer Science and Informatics, De Montfort University
The Gateway, Leicester LE1 9BH, United Kingdom
{mmavrovouniotis,syang}@dmu.ac.uk

Abstract. The population-based incremental learning (PBIL) algorithm is a combination of evolutionary optimization and competitive learning. The integration of PBIL with associative memory schemes has been successfully applied to solve dynamic optimization problems (DOPs). The best sample together with its probability vector are stored and reused to generate the samples when an environmental change occurs. It is straight forward that these methods are suitable for dynamic environments that are guaranteed to reappear, known as cyclic DOPs. In this paper, direct memory schemes are integrated to the PBIL where only the sample is stored and reused directly to the current samples. Based on a series of cyclic dynamic test problems, experiments are conducted to compare PBILs with the two types of memory schemes. The experimental results show that one specific direct memory scheme, where memory-based immigrants are generated, always improves the performance of PBIL. Finally, the memory-based immigrant PBIL is compared with other peer algorithms and shows promising performance.

1 Introduction

Population based incremental learning (PBIL) is an abstraction of evolutionary algorithms (EAs), which combines evolutionary optimization with competitive learning [1]. More precisely, PBIL explicitly maintains the statistics contained in an EA's population. Similarly to EAs, PBIL algorithms have been successfully applied to different benchmark problems and real-world applications [2–5]. In most cases, PBILs are applied to stationary optimization problems. However, many real-world problems have a dynamic environment in which the objective function, decision variables, problem instance, constraints, and so on, may vary over time [6]. Dynamic optimization problems (DOPs) are often more challenging to address because the moving optimum needs to be tracked.

Similarly with EAs, PBIL faces the same serious challenge as EAs when addressing DOPs, i.e., premature convergence. In fact, it has been confirmed that PBIL maintains significantly lower diversity than an EA does [3]. Different

strategies taken from EAs have been integrated into PBILs to address DOPs, including memory schemes [3], hyper-learning scheme [4], multi-population schemes [7] and immigrants schemes [5, 8].

In this paper, we focus on DOPs that are subject to cyclic environments, where the environments are guaranteed to re-apper in the future. Many real-world situations have cyclic or approximately cyclic dynamic environments. For example, the traffic jams in the road system are more likely to re-appear during the day. Associative memory schemes have been found suitable for cyclic DOPs, where the best samples associated with environmental information of previously optimized environments are stored and reused when the relevant environments re-appear [3, 9]. In this paper, we integrate direct memory schemes with PBIL and investigate their effect. Their difference is that direct memory schemes store only the best samples and reuse them directly to the samples whereas associative memory schemes have an indirect impact on the samples generated.

Using the exclusive-or (XOR) DOP generator that constructs cyclic DOPs proposed in [3], a series of DOPs are systematically constructed as the dynamic test environments and experiments are carried out to investigate the performance of PBILs with direct memory schemes. Based on the experimental results, the effect of direct memory schemes on the performance of PBILs in dynamic environments is analyzed. Among the direct memory schemes analyzed in this paper, the memory-based immigrants schemes have the best performance in almost all test cases. The specific algorithm is then compared with other peer PBILs and EAs and shows competitive performance.

The rest of the paper is organized as follows. Section 2 introduces the standard PBIL algorithm. Section 3 describes the existing PBIL with associative memory schemes and the proposed PBILs with direct memory schemes. Section 4 describes the construction of cyclic DOPs used for this study. The experimental study is presented in Section 5. Finally, Section 6 concludes this paper with several observations and discusses relevant future work.

2 Population-Based Incremental Learning

The standard PBIL (SPBIL) algorithm, first proposed by Baluja [1], is a combination of evolutionary optimization and competitive learning. The aim of SPBIL is to generate a real-valued probability vector $\boldsymbol{P}(t) = \{P_1, \ldots, P_l\}$ (l is the binary encoding length), at each generation t, which creates high quality solutions with high probability when sampled. Each element $P_i (i = 1, \ldots, l)$ in the probability vector is the probability of creating an allele "1" in locus i. More precisely, a solution is sampled from the probability vector $\boldsymbol{P}(t)$ as follows: for each locus i, if a randomly generated number $R \in \{0, 1\} < P_i$, it is set to 1; otherwise, it is set to 0.

SPBIL starts from an initial (central) probability vector $\boldsymbol{P}(0)$ with values of each entry set to 0.5. This means when sampling by this initial probability vector random solutions are created because the probability of generating a "1" or "0" on each locus is equal. However, as the search progresses, the values

in the probability vector are gradually moved towards values representing high evaluation solutions. The evolution process is described as follows.

For every generation t, a set $S(t)$ of n samples (solutions) are created according to the current probability vector $\boldsymbol{P}(t)$. The set of samples are evaluated according to the problem-specific fitness function. Then, the probability vector is moved towards the best solution $\boldsymbol{x}^{bs}(t)$ of the set $S(t)$ as follows:

$$P_i(t+1) \leftarrow (1-\alpha) \times P_i(t) + \alpha \times \boldsymbol{x}^{bs}(t), \quad i = \{1, \ldots, l\}, \tag{1}$$

where α is the learning rate, which determines the distance the probability vector is moved for each generation.

After the probability vector is updated toward the best sample, in order to maintain the diversity of sampling, it may undergo a bit-wise mutation process [10]. Mutation is applied to the SPBIL studied in this paper since diversity maintenance is important when addressing DOPs [3]. The mutation operation always changes the probability vector toward the central probability vector, where values are set to 0.5, to increase exploration. The mutation operation is carried out as follows. The probability of each locus P_i is mutated, if a random number $R \in \{0, 1\} < p_m$ (p_m is the mutation probability), as follows:

$$P_i'(t) = \begin{cases} P_i(t) \times (1-\delta_m), & \text{if } P_i(t) > 0.5, \\ P_i(t) \times (1-\delta_m) + \delta_m, & \text{if } P_i(t) < 0.5, \\ P_i(t), & \text{otherwise,} \end{cases} \tag{2}$$

where δ_m is the mutation shift that controls the amount a mutation operation alters the value in each bit position. After the mutation operation, a new set of samples is generated by the new probability vector and this cycle is repeated.

As the search progresses, the entries in the probability vector move away from their initial settings of 0.5 towards either 0.0 or 1.0. The search progress stops when some termination condition is satisfied, e.g., the maximum allowable number of generations is reached or the probability vector is converged to either 0.0 or 1.0 for each bit position.

PBIL has been applied to many optimization problems with promising results [2]. Most of these applications assume stationary environments, whereas only a few applications considered dynamic environments. To address DOPs with PBILs, the algorithm needs to be enhanced to maintain diversity. Existing strategies, which have been integrated with PBILs and were mainly inspired by EAs, include associative memory schemes [3], hyper-learning schemes [4], multi-population schemes [7] and immigrants schemes [8, 5]. In this paper, we integrate direct memory schemes, previously used to genetic algorithms (GAs) [11], to PBILs to address cyclically changing DOPs.

3 Memory Schemes

3.1 PBIL with Associative Memory

Memory schemes have proved to be useful in dynamic environments, especially when the environment changes cyclically. The stored information can be reused

when the environment cycles to previously optimized environments. An associative memory scheme was integrated to PBIL in [3] that stores good solutions associated with environmental information.

Within the memory-enhanced PBIL (MPBIL) a memory of size $m = 0.1 \times n$ (n is the population size) is used to store pairs of a sample with its associated probability vector. The associative memory is initially empty and stores pairs until it is full. The update policy of the memory (when it is full) is based on the closest replacement strategy: the memory point with its sample $\boldsymbol{B}_M(t)$ closest to the best population sample of the current generation $\boldsymbol{B}(t)$ in terms of the Hamming distance is replaced if it has the worst fitness; otherwise, the memory remains unchanged. Every time the memory is updated with the sample $\boldsymbol{B}(t)$ its probability vector $\boldsymbol{P}(t)$ is also stored in the memory. The memory is updated at a specific random time as follows. At the end of every memory update at generation t, a random number $R \in [5, 10]$ is generated to determine the next memory update time $t_M = t + R$.

The samples in the memory are re-evaluated every iteration. If at least one sample has a change in its fitness, then a change has occurred to the environment. Therefore, the best re-evaluated memory sample with its associated probability vector will replace the current probability vector if the memory sample has better fitness than the current best sample of the current probability vector. If no environmental change is detected, MPBIL progresses as the SPBIL does.

3.2 PBIL with Direct Memory

Direct memory schemes have showed promising performance in DOPs when applied to different GAs [11]. The difference of a direct memory scheme from the associative memory scheme described previously for the MPBIL is that the memory point consists of only the solution. In this paper, we integrate a direct memory scheme into PBIL and introduce three algorithmic variations.

Memory-Enhanced PBIL The direct memory of the memory-enhanced PBIL (MEPBIL) is initialized with random points (solutions). Similarly with MPBIL, the current best sample $\boldsymbol{B}(t)$ replaces the closest memory point $\boldsymbol{B}_M(t)$ (if no randomly initialized point still exists) if its fitness is better; otherwise, the memory remains unchanged. A memory update occurs based on the stochastic time t_M as described previously in the MPBIL. In addition, the memory is updated whenever a dynamic change occurs. In this way, the most relevant information for the specific environment will be stored in the memory.

When a dynamic change is detected the current probability vector $\boldsymbol{P}(t)$ is not replaced by the best memory probability vector as in MPBIL but the samples stored in the memory are re-evaluated and merged with current samples $S(t)$. The best $n - m$ samples will survive in the current $S(t)$. In this way, the probability vector will be learned toward $\boldsymbol{B}(t)$ that is most probably provided by a sample that was stored in the memory. When a dynamic change is not detected, the proposed MEPBIL progresses as SPBIL does.

Memory-Enhanced PBIL+Random Immigrants Random immigrants were previously integrated to PBIL to enhance their performance in (especially severely changing) DOPs [8]. Random immigrants have the ability to maintain the diversity. Since memory schemes are suitable for slightly changing environments and random immigrants are suitable for severely changing environments, a straightforward combination of MEPBIL with random immigrants may combine the merits of both.

The memory-enhanced random immigrant PBIL (MRIPBIL) differs from MEPBIL in only the additional immigration process. The immigration process within MRIPBIL occurs after the probability vector is sampled. More precisely, in every generation, $r_i \times n$ immigrants are generated and replace the worst samples in the current set of samples $S(t)$, where r_i is the replacement ratio and n is the population size. In this way, the generated immigrants will maintain diversity within the samples generated.

Memory-Based Immigrants PBIL Based on the above consideration, the memory-based immigrants scheme, which has been initially proposed for GAs [11] to address cyclic DOPs, can be integrated with PBILs, denoted as memory-based immigrants PBIL (MIPBIL) in this paper, to better tackle DOPs with reappearing environments. A direct memory scheme is maintained and updated as described previously for the MEPBIL and MRIPBIL algorithms.

Similarly with MRIPBIL above, MIPBIL also includes an immigration process but generates memory-based immigrants instead of random ones. Differently from both MRIPBIL and MEPBIL, MIPBIL does not merge the memory with the current samples but generates memory-based immigrants on every iteration as follows. For each generation t, before the mutation operation described in Eq. (2), the best memory point $\boldsymbol{B}_M(t)$ is retrieved and used as the base to generate immigrants. In every generation, $r_i \times n$ memory-based immigrants are generated by mutating bitwise with a probability p_m^i. The generated immigrants replace the worst individuals in the current set of samples $S(t)$. In this way, the samples of the next set $S(t+1)$ will have more direction toward the best point retrieved from the memory.

4 Dynamic Test Environments

The XOR DOP generator [3, 12] can construct dynamic environments from any binary-encoded stationary function $f(\boldsymbol{x})(\boldsymbol{x} \in \{0,1\}^l)$ by a bitwise XOR operator. Since for the specific experimental study memory schemes involved cyclic DOPs are considered. With the XOR DOP, the cyclicity of the changing environments is controlled as follows. Initially, $2K$ XOR masks $\boldsymbol{M}(0), \ldots, \boldsymbol{M}(2K-1)$ are generated as the base states in the search space randomly. Then, the environment can cycle among these base states in a fixed logical ring. Suppose the environment changes in every τ algorithmic generations, then the individuals at generation t are evaluated as follows:

$$f(\boldsymbol{x}, t) = f(\boldsymbol{x} \oplus \boldsymbol{M}(I_t)) = f(\boldsymbol{x} \oplus \boldsymbol{M}(k\%(2K))), \tag{3}$$

where "$\oplus$" is the XOR operator (i.e., $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$), $k = \lceil t/f \rceil$ is the index of the current environment, $I_t = k\%(2K)$ is the index of the base state that the environment is in at generation t and M are the XORing masks.

The $2K$ XORing masks are generated as follows. First, K binary templates $\boldsymbol{T}(0), \ldots, \boldsymbol{T}(K-1)$ are constructed with each template containing exactly $\rho \times n = l/K$ ones. Let $\boldsymbol{M}(0) = 0$ denote the initial state. Then, the other XORing masks are generated iteratively as follows:

$$\boldsymbol{M}(i+1) = \boldsymbol{M}(i) \oplus \boldsymbol{T}(i\%K),\ i = 0, \ldots, 2K-1. \tag{4}$$

The templates $\boldsymbol{T}(0), \ldots, T(K-1)$ are first used to create K masks till $\boldsymbol{M}(K) = 1$ and then orderly reused to construct another K XORing masks till $\boldsymbol{M}(2K) = \boldsymbol{M}(0) = 0$. The Hamming distance between two neighbor XOR masks is the same and equals $\rho \times n$. Here, $\rho \in [1/l, 1.0]$ is the distance factor, determining the number of base states. A higher value of ρ means severer dynamic changes, whereas a lower value of τ means faster dynamic changes.

In this paper, four 100-bit binary-encoded problems are selected as the stationary problems to generate DOPs. Each problem consists of 25 copies of 4-bit building blocks and has an optimum of 100. The first one is the *OneMax* function, which aims to maximize the number of ones in a solution. The second one is the *Plateau* function, where each building block contributes four (or two) to the total fitness if its unitation (i.e., the number of ones inside the building block) is four (or three); otherwise, it contributes zero. The third one is the *RoyalRoad* function, where each building block contributes four to the total fitness if its unitation is four; otherwise, it contributes zero. The fourth one is the *Deceptive* function, where the building block is a fully deceptive sub-function. Generally, the difficulty of the four functions for optimization algorithms is increasing in the order from OneMax to Plateau to RoyalRoad to Deceptive.

5 Experimental Study

5.1 Experimental Setup

For each algorithm on a DOP, 30 independent runs were executed on the same environmental changes and 100 environmental changes were allowed for each run. The overall performance is calculated as follows:

$$\bar{F}_{BOG} = \frac{1}{G} \sum_{i=1}^{G} \left(\frac{1}{N} \sum_{j=1}^{N} F_{BOG_{ij}} \right), \tag{5}$$

where G is the number of generations of a run, N is the number of runs and $F_{BOG_{ij}}$ is the best of generation fitness of generation i of run j. Moreover, the diversity of the population was recorded every generation. The overall diversity of an algorithm on a DOP is defined as:

$$\bar{T}_{DIV} = \frac{1}{G} \sum_{i=1}^{G} \left(\frac{1}{N} \sum_{j=1}^{N} Div_{ij} \right), \tag{6}$$

where G and N are defined as in Eq. (5) and Div_{ij} is the diversity at generation i of run j, which is defined as:

$$Div_{ij} = \frac{1}{ln(n-1)} \sum_{p=1}^{n} \sum_{q \neq p}^{n} HD(p,q), \tag{7}$$

where l is the encoding length, n is the population size and $HD(p,q)$ is the Hamming distance between the p-th sample and q-th sample.

Dynamic test environments are generated from the four aforementioned binary-encoded functions, described in Section 4, using the XOR DOP generator with τ set to 10 and 50, indicating quickly and slowly changing environments, respectively, and ρ set to 0.1, 0.2, 0.5 and 1.0, indicating slightly, to medium, to severely changing environments, respectively. With the specific setting of ρ, basically the environments cycles among 20, 10, 4, and 2 base states respectively. As a result, eight cyclic dynamic environments (i.e., 2 values of $\tau \times 4$ values of ρ) from each stationary function are generated to systematically analyze the algorithms on the DOPs.

In order to have fair comparisons among PBILs, the population size, memory size and immigrant replacement are set to such that each PBIL has 120 fitness evaluation per generation. For memory-based algorithms the memory size was set to $m = 0.1 \times n$ and for all immigrant-based algorithms the immigrant replacement ratio was set to $r_i = 0.2$ and immigrant mutation probability $p_m^i = 0.05$. For all PBILs in the experiments, the parameters were set to typical values [3] as follows: the learning rate $\alpha = 0.25$, mutation probability $p_m = 0.05$ with the mutation shift $\delta_m = 0.05$, and the elitism of size 1.

5.2 Experimental Study of Memory-Based PBILs

The experimental results of the proposed PBILs with direct memory schemes (e.q., MEPBIL, MRIPBIL and MIPBIL) compared with the existing PBIL with associative memory scheme (e.g., MPBIL) are shown in Fig. 1. The corresponding statistical results are presented in Table 1, where Kruskal–Wallis tests were applied followed by posthoc paired comparisons using Mann–Whitney tests with the Bonferroni correction. The statistical results are shown as "$-$", "$+$" or "$\sim$" when the first algorithm is significantly better than the second algorithm, when the second algorithm is significantly better than the first algorithm, or when the two algorithms are insignificantly different, respectively. To better understand the behaviour of the PBILs the population diversity against generations for the first 500 generations is plotted in Fig. 2 on the DOPs with $\tau = 50$ and $\rho = 0.2$. From Fig. 1 and Table 1, the following observations can be drawn.

First, among the PBILs with direct memory schmes, MIPBIL has the best performance and outperforms both MEPBIL and MRIPBIL in most DOPs. When random immigrants are hybridized with memory schemes the performance is enhanced only in DOPs with $\rho = 1.0$ but degraded in the remaining DOPs; see the comparisons of MEPBIL $\Leftrightarrow$ MRIPBIL in Table 1. This is normal because random immigrants generate high levels of diversity that help the adaptation

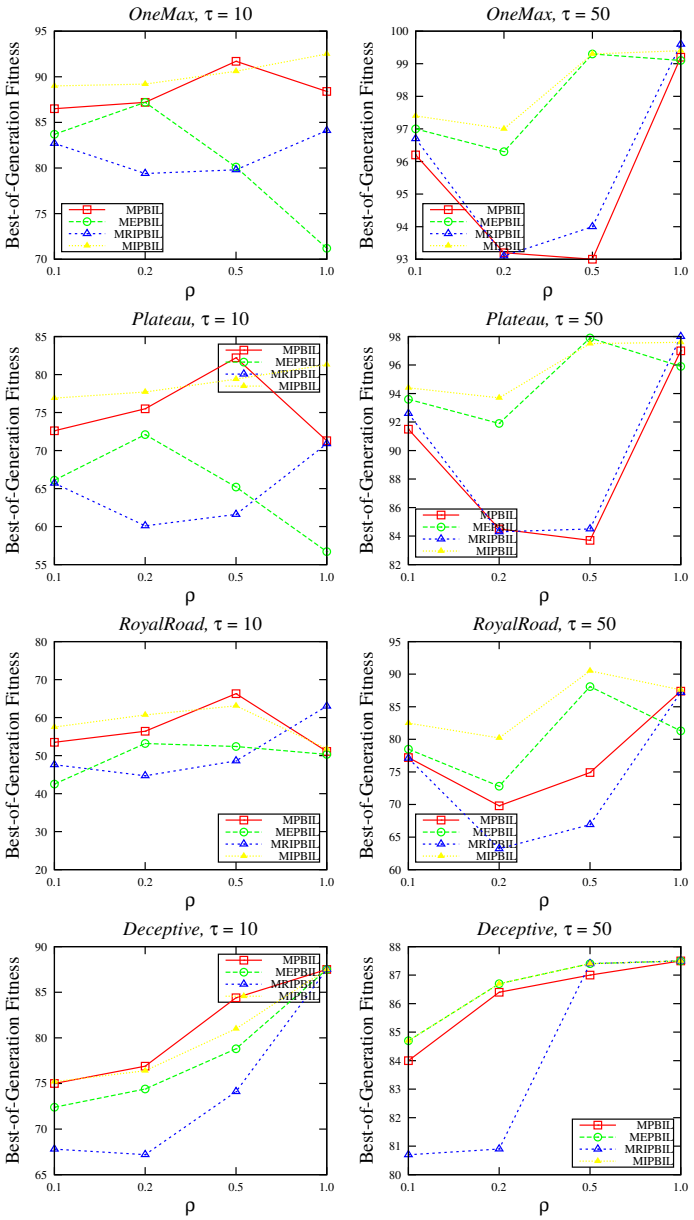

Fig. 1. Offline performance of memory-based PBILs for DOPs

in severely changing environments but may disturb the optimization process in slightly changing environments. In contrast, memory-based immigrants in MIPBIL generate guided diversity via transferring knowledge from previous environ-

Table 1. Statistical results of comparing memory-based PBILs on cylic DOPs

	$\tau = 10$				$\tau = 50$			
DOPs, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
Algorithms	*OneMax*							
MPBIL ⇔ MEPBIL	−	∼	−	−	+	+	+	−
MPBIL ⇔ MRIPBIL	−	−	−	−	+	∼	+	+
MPBIL ⇔ MIPBIL	+	+	−	∼	+	+	+	+
MIPBIL ⇔ MEPBIL	−	−	−	−	−	−	−	−
MIPBIL ⇔ MRIPBIL	−	−	−	−	−	−	−	+
MEPBIL ⇔ MRIPBIL	−	−	∼	+	−	−	−	+
Algorithms	*Plateau*							
MPBIL ⇔ MEPBIL	−	−	−	−	+	+	+	−
MPBIL ⇔ MRIPBIL	−	−	−	∼	+	∼	+	+
MPBIL ⇔ MIPBIL	+	+	−	+	+	+	+	+
MIPBIL ⇔ MEPBIL	−	−	−	−	−	−	∼	−
MIPBIL ⇔ MRIPBIL	−	−	−	−	−	−	−	+
MEPBIL ⇔ MRIPBIL	∼	−	−	+	−	−	−	+
Algorithms	*RoyalRoad*							
MPBIL ⇔ MEPBIL	−	−	−	∼	∼	+	+	−
MPBIL ⇔ MRIPBIL	−	−	−	+	∼	−	−	∼
MPBIL ⇔ MIPBIL	+	+	−	+	+	+	+	∼
MIPBIL ⇔ MEPBIL	−	−	−	−	−	−	−	−
MIPBIL ⇔ MRIPBIL	−	−	−	+	−	−	−	∼
MEPBIL ⇔ MRIPBIL	+	−	−	+	−	−	−	+
Algorithms	*Deceptive*							
MPBIL ⇔ MEPBIL	−	−	−	∼	+	∼	∼	∼
MPBIL ⇔ MRIPBIL	−	−	−	∼	−	−	∼	∼
MPBIL ⇔ MIPBIL	∼	∼	−	∼	+	∼	∼	∼
MIPBIL ⇔ MEPBIL	−	−	−	∼	∼	∼	∼	∼
MIPBIL ⇔ MRIPBIL	−	−	−	∼	−	−	∼	∼
MEPBIL ⇔ MRIPBIL	−	−	−	∼	−	−	∼	∼

ments. Fig. 2 supports our claim since MRIPBIL maintains a higher diversity level than MIPBIL.

Second, MEPBIL and MRIPBIL outperform MPBIL in most DOPs with $\tau = 50$ whereas MPBIL outperfoms MEPBIL and MRIPBIL in most DOPs with $\tau = 10$; see the comparisons in Table 1. In contrast, MIPBIL outperforms MPBIL in most DOPs; see the comparisons of MPBIL ⇔ MIPBIL in Table 1. From Fig. 2 it can be observed that MIPBIL maintains higher diversity levels than MPBIL especially in DOPs with $\tau = 10$. Low diversity levels show that MPBIL possibly loses its adaptation capabilities and cannot track the cyclic changing environments when they change fast.

Since the memory-based immigrants scheme has the best performance over the other direct memory schemes, we only consider MIPBIL for the remaining experiments.

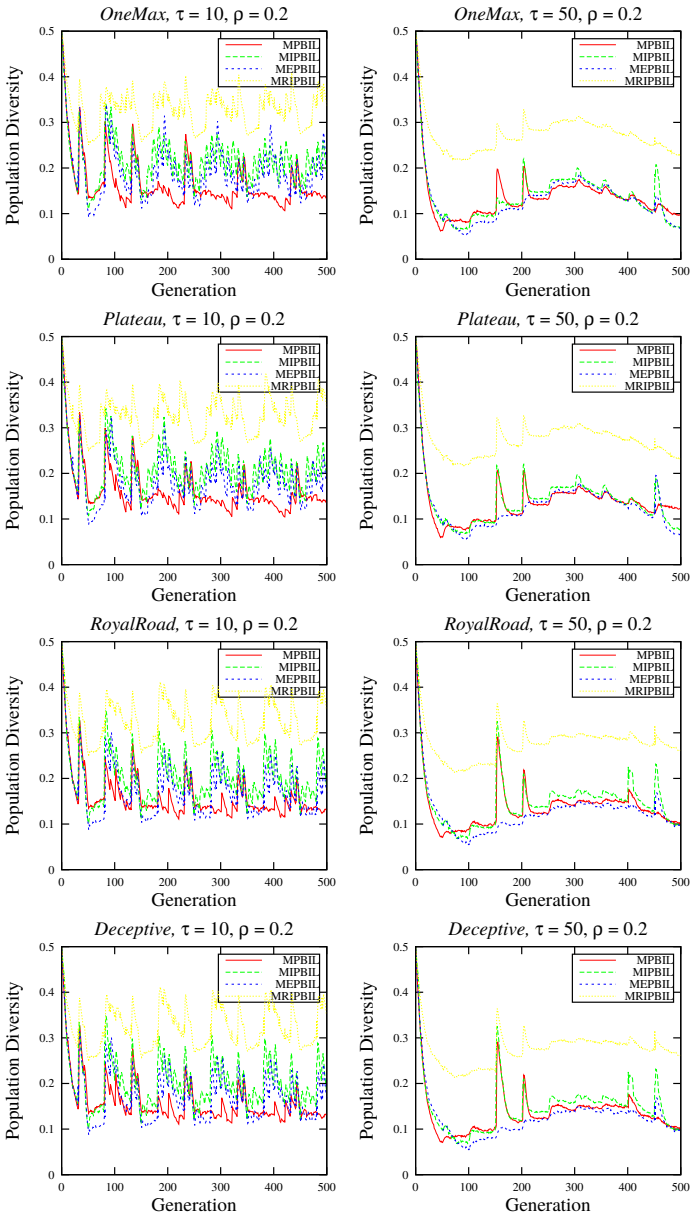

Fig. 2. Dynamic population diversity of PBILs for the first five hundred generations on DOPs with $\tau = 50$ and $\rho = 0.2$

5.3 Experimental Study of MIPBIL with Other PBILs

The best performing PBIL with the direct memory scheme, i.e., MIPBIL is compared with other existing PBILs: SPBIL [2], random immigrants PBIL (RIPBIL)

Table 2. Offline performance results of MIPBIL against other PBILs algorithms on cyclic DOPs.

	$\tau = 10$				$\tau = 50$			
DOPs, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
Algorithms	*OneMax*							
SPBIL	79.6	68.0	59.4	57.1	96.8	92.8	81.8	70.0
RIPBIL	79.3	75.5	74.8	80.7	96.7	92.6	89.0	92.2
EIPBIL	**90.1**	77.7	65.0	61.2	**98.5**	96.4	88.9	80.3
HIPBIL	88.1	77.1	75.1	82.9	98.1	95.7	90.7	94.0
MIPBIL	89.0	**89.2**	**90.6**	**92.5**	97.4	**97.0**	**99.3**	**99.4**
Algorithms	*Plateau*							
SPBIL	60.5	47.0	43.3	50.5	93.0	84.0	61.4	53.6
RIPBIL	60.7	55.4	55.1	66.5	92.7	83.4	76.3	83.9
EIPBIL	**78.0**	58.1	47.3	50.7	**96.4**	91.4	72.7	55.5
HIPBIL	74.2	59.2	56.9	71.4	95.7	89.9	78.4	87.4
MIPBIL	76.9	**77.7**	**79.4**	**81.3**	94.4	**93.7**	**97.5**	**97.6**
Algorithms	*RoyalRoad*							
SPBIL	42.6	34.4	37.0	50.0	78.6	59.6	43.8	50.3
RIPBIL	45.1	40.5	43.9	59.2	77.7	60.7	59.5	72.7
EIPBIL	56.3	41.0	39.6	50.0	**85.8**	68.0	47.7	50.4
HIPBIL	52.6	44.3	46.4	**64.8**	84.3	66.2	60.5	76.6
MIPBIL	**57.5**	**60.7**	**63.1**	51.8	82.5	**80.2**	**90.5**	**87.6**
Algorithms	*Deceptive*							
SPBIL	67.1	64.9	69.6	87.5	78.2	75.3	76.9	87.5
RIPBIL	66.7	64.9	70.7	87.5	78.0	75.1	79.2	87.5
EIPBIL	72.6	69.2	72.7	87.5	80.1	78.5	80.7	87.5
HIPBIL	72.6	67.9	76.5	**97.2**	77.5	74.0	82.9	**98.9**
MIPBIL	**75.2**	**76.4**	**81.0**	87.5	**84.7**	**86.7**	**87.4**	87.5

[3], elitism-based PBIL (EIPBIL) [8] and hybrid immigrants PBIL (HIPBIL) [8]. The experimental results of the aforementioned algorithms are presented in Table 2. Bold values indicate that the algorithm is significantly better than the other algorithms using the same statistical method described previously.

From Table 2, it can be observed that MIPBIL outperforms SPBIL in all DOPs except some cases of the Deceptive function. Also, MIPBIL outperforms the remaining algorithms in most DOPs. This is because MIPBIL can directly move the population to the optimum or close to the optimum of previously optimized environments. Specifically, MIPBIL is underperformed by EIPBIL only in most DOPs with $\rho = 0.1$. This is because when $\rho = 0.1$ more base states (i.e., 20) exist and the memory points stored may not be so accurate to the relevant base states and may misguide the immigrants. As ρ increases the performance of MIPBIL improves whereas EIPBIL degrades which further supports our claim. MIPBIL is underperformed by HIPBIL only in some DOPs with $\rho = 1.0$. This is because when $\rho = 1.0$ the environments switches between two fitness landscapes that are complementary to each other. Therefore, the dualism type of

Table 3. Offline performance of MIPBIL against MIGA on cylic DOPs.

	$\tau = 10$				$\tau = 50$			
DOPs, $\rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
Algorithms	*OneMax*							
MIPBIL (vs)	**89.0**	**89.2**	90.6	92.5	**97.4**	**97.0**	**99.3**	**99.4**
MIGA	82.1	83.5	**92.7**	**96.0**	95.9	93.6	98.4	99.3
Algorithms	*Plateau*							
MIPBIL	**76.9**	**77.7**	79.4	81.3	**94.4**	**93.7**	**97.5**	**97.6**
MIGA	65.4	67.3	**82.3**	**89.7**	91.4	86.5	95.3	**97.6**
Algorithms	*RoyalRoad*							
MIPBIL (vs)	**57.5**	**60.7**	**63.1**	51.8	**82.5**	**80.2**	**90.5**	**87.6**
MIGA	45.2	49.1	61.3	**65.4**	75.7	69.9	83.7	**86.9**
Algorithms	*Deceptive*							
MIPBIL (vs)	**75.2**	**76.4**	81.0	**87.5**	**84.7**	**86.7**	**87.4**	**87.5**
MIGA	66.0	73.4	**82.5**	86.0	81.3	85.2	86.9	87.2

immigrants (e.g., complementary to the best solution) generated in HIPBIL are suitable to cope with these types of DOPs.

5.4 Experimental Results on Pairwise Comparisons of MIPBIL with MIGA

Since the direct memory scheme, i.e., memory-based immigrants, integrated to MIPBIL, was initially introduced and integrated to GAs [11], further pairwise comparisons are performed in this section. Specifically, MIPBIL is compared with memory-based immigrants GA (MIGA) [11]. MIGA is executed on the same DOPs and common parameter settings with MIPBIL above (e.g., $r_i = 0.2$, $p_m^i = 0.05$ and $m = 0.1 \times n$). The remaining MIGA parameters were set to typical values as follows: generational, uniform crossover with $p_c = 0.6$, flip mutation with $p_m = 0.01$, and fitness proportionate selection with elitism of size 1. The pairwise comparisons regarding the performance are given in Table 3. A bold value indicates that the algorithm is significantly better than the other using Mann–Whitney tests. The dynamic performance of the two algorithms with respect to the best-of-generation fitness against generation on the DOPs with $\tau = 50$ and $\rho = 0.2$ is plotted in Fig. 3.

From Table 3, it can be observed that MIPBIL outperforms MIGA on all DOPs with $\tau = 50$ and on most DOPs with $\tau = 10$ and $\rho = 0.1$ and $\rho = 0.2$. MIGA outperforms MIPBIL in DOPs with $\tau = 10$ and $\rho = 0.5$ and $\rho = 1.0$. This is probably because the mutation operator of MIGA may have a faster effect than the mutation operator of MIPBIL to the adaptation process of the algorithm. The difference lies in that the former is direct (applied to the solutions) whereas the latter is indirect (applied to the probabilistic vector) and may need some time to express its effect.

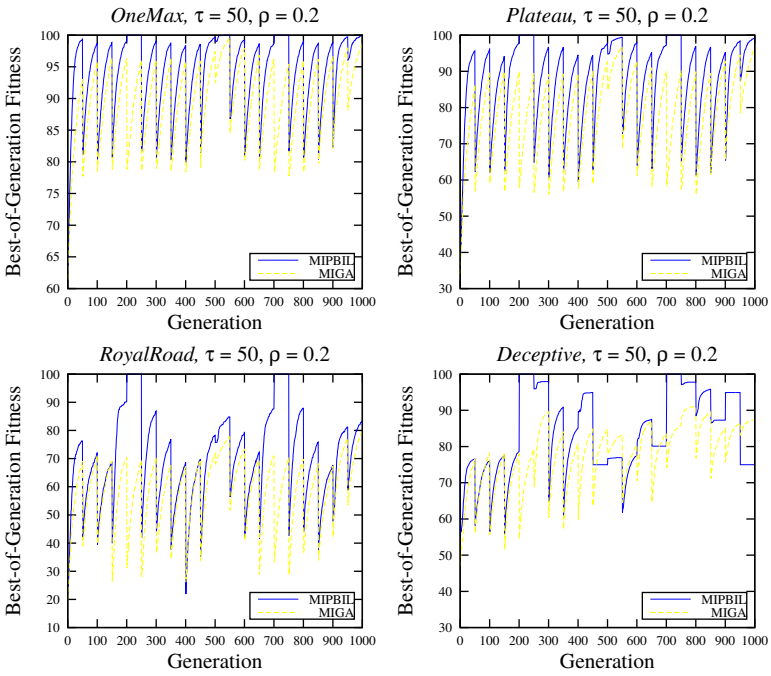

Fig. 3. Dynamic performance of MIPBIL and MIGA for the first twenty environmental changes on DOPs with $\tau = 50$ and $\rho = 0.2$.

6 Conclusions

Associative memory schemes have been successfully integrated with PBILs [3]. In this paper, three direct memory schemes are integrated with PBIL and their performance is investigated on cyclically changing DOPs. The direct memory scheme in which memory-based immigrants are generated has the best performance when integrated with PBIL. Therefore, its performance is further compared against other peer algorithms.

From the experimental results, the following concluding remarks can be drawn. First, among the direct schemes integrated with PBIL, the memory-based immigrants scheme has the best performance. Second, although associative memory schemes contain more environmental information from direct memory schemes, the latter may provide faster adaptation in PBILs. Third, PBILs may also benefit when integrated with memory-based immigrants as with GAs [11]. In fact from the experiments it can be observed that MIPBIL outperforms MIGA in many DOPs.

For future work, it would be interesting to apply PBILs to the Knapsack problem that has many applications in the real world.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/K001310/1.

References

1. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, USA (1994)
2. Larrañaga, P., Lozano, J., eds.: Estimation of distribution algorithms: a new tool for evolutionary computation. Kluwer, Norwell, MA (2002)
3. Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Transactions on Evolutionary Computation **12**(5) (2008) 542–561
4. Yang, S., Richter, H.: Hyper-learning for population-based incremental learning in dynamic environments. In: Evolutionary Computation, 2009. CEC '09. IEEE Congress on. (2009) 682–689
5. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Computing **9**(11) (2005) 815–834
6. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments–a survey. IEEE Transactions on Evolutionary Computation **9**(3) (2005) 303–317
7. Yang, S.: Population-based incremental learning with memory scheme for changing environments. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. GECCO '05, New York, NY, USA, ACM (2005) 711–718
8. Mavrovouniotis, M., Yang, S.: Population-based incremental learning with immigrants schemes for changing environments. In: Proceedings of the 2015 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE). (Dec 2015) 1444–1451
9. Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J., Romero, J., Smith, G., Squillero, G., Takagi, H., eds.: Applications of Evolutionary Computing. Volume 3907 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2006) 788–799
10. Baluja, S.: An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University, Pittsburgh, PA, USA (1995)
11. Yang, S.: Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. Evolutionary Computation **16**(3) (2008) 385–416
12. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm. In: The 2003 Congress on Evolutionary Computation CEC '03. Volume 3. (Dec 2003) 2246–2253