Soft Comput (2011) 15:311–326
DOI 10.1007/s00500-010-0547-5

ORIGINAL PAPER

Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments

Xingguang Peng · Xiaoguang Gao · Shengxiang Yang

Published online: 11 February 2010

Abstract In estimation of distribution algorithms (EDAs), the joint probability distribution of high-performance solutions is presented by a probability model. This means that the priority search areas of the solution space are characterized by the probability model. From this point of view, an environment identification-based memory management scheme (EI-MMS) is proposed to adapt binary-coded EDAs to solve dynamic optimization problems (DOPs). Within this scheme, the probability models that characterize the search space of the changing environment are stored and retrieved to adapt EDAs according to environmental changes. A diversity loss correction scheme and a boundary correction scheme are combined to counteract the diversity loss during the static evolutionary process of each environment. Experimental results show the validity of the EI-MMS and indicate that the EI-MMS can be applied to any binary-coded EDAs. In comparison with three state-of-the-art algorithms, the univariate marginal distribution algorithm (UMDA) using the EI-MMS performs better when solving three decomposable DOPs. In order to understand the EI-MMS more deeply, the sensitivity analysis of parameters is also carried out in this paper.

X. Peng (✉) · X. Gao
School of Electronics and Information, Northwestern Polytechnical University, Xi'an, Shaanxi 710129, China
e-mail: pxg0510@gmail.com

X. Gao
e-mail: xggao@nwpu.edu.cn

S. Yang
Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH, UK
e-mail: s.yang@mcs.le.ac.uk

Keywords Estimation of distribution algorithm · Dynamic optimization problem · Environment identification · Memory scheme · Diversity compensation

1 Introduction

In the real world, optimization problems are usually time-varying and it is very important to get the optimum in a short and acceptable time. Many researchers have contributed to this challenging issue of solving dynamic optimization problems (DOPs). Evolutionary algorithms (EAs) are inspired by the evolutionary process in nature. From the evolutionism point of view, the nature process simulated by EAs is changing, random, and uncertain in itself. Therefore, it is very reasonable to use EAs to solve DOPs. The simplest way to react to an environmental change is to regard each change as the arrival of a new optimization problem, and solve it from scratch. However, the time between every two environmental changes is usually rather short in most DOPs. Hence, the restart approach cannot satisfy most of real-world DOPs. In recent years, researchers have developed many methods to maintain a sufficient diversity level for EAs to continuously adapt to the changing landscape. They can be classified into four categories (Jin and Branke 2005): (1) generating diversity after a change, such as the hyper-mutation method (Cobb 1990); (2) maintaining the diversity throughout the run, such as the random immigrants (Grefenstette and Fitzpatrick 1992), sharing or crowding mechanisms (Cedeno and Vemuri 1997), and the thermodynamical genetic algorithm (GA) (Mori et al. 1996); (3) memory-based approaches (Branke 1999), and (4) multi-population approaches, such as the

self-organizing scouts GA (Branke et al. 2000), the multi-national GA (Ursem 2000), and the shift balance GA (Wineberg and Oppacher 2000). Comprehensive surveys on EAs applied to dynamic environments can be found in (Branke 2001; Jin and Branke 2005; Morrison 2004).

The essence of DOPs is to search the optimum in the solution space dynamically. For such a dynamic process, the historic information generated in the previous search process is very useful. An intuitional method is to store the high-performance historic solutions and reuse them later so as to improve the search process. But this method involves a large memory space and a complex memory management scheme. Estimation of distribution algorithms (EDAs) are a class of probability model based EAs, where the processes of learning and sampling the probability model replace the genetic operations (e.g., crossover and mutation) in conventional GAs. A probability model indicates the joint probability distribution of high-performance solutions. That is, it characterizes the set of good solutions. If the historic information could be stored as probability models, we would not only save the memory space but also simplify the memory management scheme. Consequently, EDAs are suitable for being extended to be memory-enhanced EAs to solve DOPs.

To this end, an environment identification-based memory management scheme (EI-MMS) is proposed in this paper. Within this scheme, a probability model is regarded as the learning result of the probability distribution of high-performance solutions in an environment. A probability model together with the best individual in the solutions from which the probability model is learnt are stored as a memory element. In order to retrieve the memory elements in EI-MMS, an environment identification method is proposed to select the suitable element according to a special environment. The EI-MMS can be used to extend any binary-encoded static EDA to its dynamic version and we name the corresponding algorithm as EDA with environment identification based memory scheme (EI-MEDA).

Considering the fact that the diversity of conventional EDAs will loss gradually while the learning and sampling processes of the probability models are executed alternately, in this paper, the reason to diversity loss is briefly analyzed and an effective diversity compensation method is introduced into EI-MEDA to enhance its performance in dynamic environments.

The rest of this paper is organized as follows. Section 2 presents the description of the proposed EI-MEDA. In Sect. 3, the diversity loss reason is first analyzed and some diversity loss counteracting methods are then introduced into EI-MEDA. Section 4 presents the experimental results and analysis. Finally, conclusions are drawn in Sect. 5.

2 Description of the EI-MEDA

In this section, we present the details about the EI-MEDA. Any static binary-coded EDA can be extended to its corresponding EI-MEDA using EI-MMS. From this point of view, an EI-MEDA is composed of two main parts: the basic EDA and EI-MMS. The former aims at searching optimum in each environment and the latter aims at adapting to the environment changes.

2.1 Introduction of EDAs

The concept of EDAs was firstly proposed in 1996 (Mühlenbein and Paaß 1996). In an EDA, the probability distribution of high-performance solutions is estimated and is used to generate new candidate solutions. There are five main steps in EDAs: selection, learning, sampling, replacement, and evaluation which is shown in Fig. 1. It can be seen that the learning and sampling steps replace the crossover and mutation operations in simple GAs.

2.2 The EI-MMS

This EI-MMS scheme uses additional memory and the elements stored are the probability models learnt from the population. Before giving the details, we assume that the environmental changes are detectable. In all the algorithms below, the environmental change is detected in each generation by checking whether there is at least one memory element whose evaluation value has changed.

In order to utilize the intervals between every two environmental changes to learn a high-quality probability model, EI-MEDA updates its memory just after the

```
begin
t := 0
Randomly initialize first population P_0
  repeat
    select promising solutions S_t from P_t (selection)
    estimate the probability distribution of S_t (learning)
    generate offspring O_t using the estimate (sampling)
    create a new population P_{t+1} by replacing some
      solutions from P_t with O_t (replacement)
    evaluate the individuals in P_{t+1} (evaluation)
    t := t + 1;
  until terminated = true;
end;
```

Fig. 1 Pseudocode for EDAs

environment changes. As shown in Fig. 2, the whole dynamic optimization process is divided into many static optimization processes. In each static process, EDA searches the optimum in its conventional way. When the eth environment comes at generation t, EI-MMS manages its memory $\mathbf{M}$ in three major steps. First, it stores the probability model obtained from the generation just before the environmental change, i.e., $\mathrm{PM}(t-1)$, into the memory. Then, it finds a memory element $M(k_e)$ $(k_e = 1, 2, \ldots, m)$ which best fits the new environment to retrieve using an environment identification method. Finally, the probability model of this memory element, i.e., $m\mathrm{PM}(k_e)$, is sampled to generate the first generation of population in the new environment.

2.3 The environment identification method

The environment identification method is very important due to its role of linking between the memory and the dynamic environment. A key aspect of EI-MMS is to find the suitable memory element to retrieve according to the new environment. An intuitive way of achieving this is to consider the average fitness of the solutions sampled from a special element. Considering the computational complexity and the accuracy, we propose a samples averaging plus best individual (SA + BI) method to evaluate the elements in the memory and select the suitable one.

2.3.1 The samples averaging (SA) method

The idea of this method is to evaluate a memory element by averaging the fitness of solutions sampled from it. For each memory element $M(k)$ $(k = 1, \ldots, m)$, N_S solutions are sampled from it and the average fitness of these sampled solutions is calculated in the current environment as the evaluation value of $M(k)$ as follows.

$$f_M(k) = \frac{1}{N_S}\sum_{i=1}^{N_S} f^k_{ind(i)} \tag{1}$$

where $f^k_{ind(i)}$ denotes the fitness of the ith solution sampled from the probability model, i.e. $m\mathrm{PM}(k)$ of $M(k)$. This method is the most intuitive way to evaluate a memory element but its computational complexity is high.

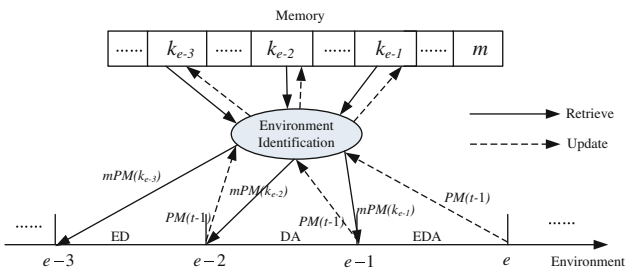

Fig. 2 Illustration of the EI-MMS

2.3.2 The best individual (BI) method

In this method, each memory element consists of two parts: a probability model and the best individual of the population from which the probability model is learnt. Here, we denote the memory element by $M(k) = \langle \mathrm{BM}(k), m\mathrm{PM}(k)\rangle$ $(k = 1, 2, \ldots, m)$, where $\mathrm{BM}(k)$ denotes the best individual. The evaluation of the memory element $M(k)$ is defined as follows:

$$f_M(k) = f(\mathrm{BM}(k)) \tag{2}$$

where $f(\mathrm{BM}(k))$ denotes the fitness of $\mathrm{BM}(k)$ in the current environment. This method is similar to the method used in (Yang 2005b, 2006).

In contrast with the SA method, the accuracy is sacrificed for the sake of the computational complexity. The BI method uses the fitness of the best individual to evaluate the probability model learning from a set of individuals. This may lead to inaccuracy. For example, it is impossible to differentiate two elements when the fitness of their best individuals is equal.

2.3.3 The SA + BI method

In order to balance the accuracy and the computational complexity, we combine the above two methods, resulting in the SA + BI method. For comparing two memory elements, if the fitness of the best individuals are different, the BI method is applied; otherwise, the SA method is applied to differentiate the memory elements. Figure 3 shows how to select a memory element with the SA + BI method, where a maximization problem is assumed. The pseudo-code of the proposed EI-MEDA with the SA + BI method is shown in Fig. 4, where N_{pop} denotes the population size and p_s denotes the truncation selection rate (i.e., for each generation, the $p_s \times N_{pop}$ best samples generated from the current model $\mathrm{PM}(t)$ are selected to build up the model $\mathrm{PM}(t+1)$ for the next generation).

3 Diversity loss and counteracting methods

The conventional EDA is likely to search the space where it has visited, just like the GA without a mutation operation. When the probability distribution of a decision variable is close to 1 or 0, it is difficult to change its value anymore. This is the so-called fixed-point problem and it may mislead the search process to a local optimum. Some researchers have contributed to this challenge (Branke et al. 2007; Shapiro 2003, 2005, 2006).

```
Input: M(i) = ⟨BM(i), mPM(i)⟩, (i = 1, 2, ..., m)
Output: The suitable memory element M(index)
  maxfit := 0 and index := 1;
  for i := 1 to m do
    Calculate f_M(i) by Eq. (2);
    if f_M(i) > maxfit then
      maxfit := f_M(i) and index := i;
    else if f_M(i) = maxfit then
      Calculate f_M(i) and f_M(index) by Eq. (1);
      if f_M(i) > f_M(index) then index := i;
  endfor;
  return M(index);
```

Fig. 3 Pseudocode for the SA + BI method

```
begin
PM(0, i) := 0.5, mPM(j, i) := 0.5, i ∈ [1, l], j ∈ [1, m];
Randomly generate BM(j), t := 0, k := 0;
Randomly initialize first population P_0;
  repeat
    if an environment change is detected then
      ⟨BM(k), mPM(k)⟩ := ⟨B(t − 1), PM(t − 1)⟩;
      Update the fitness value of each BM(j), j ∈ [1, m];
      Select a proper mPM(k) using the SA+BI method;
      PM(t) := mPM(k);
      Sample PM(t) to generate P_t;
    else
      Evaluate P_t;
      Select the p_s × N_pop best individuals from P_t
      Learn PM(t);
      Compensate the diversity for PM(t) by the method
        described in Section 3;
      Sample PM(t) to generate the t-th offspring O_t;
      Evaluate O_t and replace the worst individuals in P_t;
      B(t) := BestIndvidual(P_t);
    t := t + 1;
  until terminated = true;
end;
```

Fig. 4 Pseudocode for the proposed EI-MEDA

3.1 The reason to the diversity loss

It is well known that the variance of a sample of size N has an expected value of $\sigma^2(1 - 1/N)$ where σ^2 is the variance in the parent distribution. Most EDAs do not compensate for this. When the new probability model is produced, it attempts to model the new population, and therefore, has a reduced variance. When this is iterated repeatedly, the variance of the sampled population gets smaller and smaller and decays to zero. The probability model evolves to one which can only generate identical configurations. In Shapiro (2005) analyzed the dynamics of EDAs in terms of Markov chains and declared that the general EDAs cannot satisfy two necessary conditions for being effective search algorithms. Hence, we must counteract the diversity loss to improve the efficiency of an EDA.

3.2 Basic diversity compensation methods

As mentioned above, EI-MMS is in fact a diversity maintaining method according to the environmental changes. It is also important to counteract the diversity loss in static EDA which searches the optimum in each environment. Here, we introduce some basic diversity compensation methods for binary EDAs. According to the experimental study in Branke et al. (2007), the method that combines the loss correction and boundary correction methods, denoted LC + BC in this paper, is outstanding to counteract the diversity loss. Hence, we use the LC + BC method as the basic diversity compensation method in EI-MEDA. The details are given below and the experimental results are shown in the next section.

3.2.1 The loss correction (LC) method

Let l be the length of a chromosome and $\gamma_i (i = 1, \ldots, l)$ be the probability that the allele of the ith gene is equal to 1, γ_i is transformed to γ_i' to counteract the diversity loss as follows:

$$\gamma_i' = \begin{cases} \frac{1-\sqrt{1-4(1-\gamma_i)/L_s}}{2}, & \gamma_i \le \frac{1}{2}(1-\sqrt{1-L_S}) \\ \frac{1+\sqrt{1-4(1-\gamma_i)/L_s}}{2}, & \gamma_i \ge \frac{1}{2}(1+\sqrt{1-L_S}) \\ 0.5, & \text{otherwise} \end{cases} \tag{3}$$

where $L_S = \frac{p_s \times N_{pop} - 1}{p_s \times N_{pop} - p_s}$.

3.2.2 The boundary correction (BC) method

For the BC method, γ_i is transformed to γ_i' to counteract the diversity loss as follows:

$$\gamma_i' = \begin{cases} \beta, & \gamma_i < \beta \\ 1-\beta, & \gamma_i > 1-\beta \\ \gamma_i, & \text{otherwise} \end{cases} \tag{4}$$

where β is a preset parameter to prevent the distribution from converging to 1 or 0. To guarantee the minimal diversity level, β is set to $1/l$ in this paper unless stated otherwise.

3.2.3 The LC + BC method

For the LC + BC method, LC and BC are applied in turn. In other words, LC is first applied to γ_i, then the resulting γ_i' is taken as the input to the BC method. As shown in Fig. 5a, at the beginning of the searching process, i.e., γ_i is close to 0.5, the effect of LC is the strongest because it always returns γ_i to 0.5. At the early searching stage, this effect that LC counteracts γ_i from evolving towards 0 or 1 enables the population search more widely in the solution space. But, LC cannot prevent the population from converging because it does not guarantee the minimal diversity level. On contrast, as shown in Fig. 5b, BC can prevent a distribution from converging by forcing the distribution with a minimal diversity level. Therefore, the combination of LC and BC, as shown in Fig. 5c, cannot only enable the algorithm to search widely but also prevent the population from converging when the distributions are close to their extreme value, i.e., 0 or 1.

4 Experimental study

4.1 Dynamic test environments and measurement

Here, we present a bitwise exclusive-or (XOR) DOP generator, which was first proposed in Yang (2003) and Yang and Yao (2005) and then finalized in Yang (2005a) and Yang and Yao (2008). This DOP generator can construct three types of dynamic environment (cyclic, cyclic with noise, and random environment) from any binary-encoded function $f(\mathbf{x}), \mathbf{x} \in \{0,1\}^l$ by an XOR operator. For each environmental period k, a XORing mask $\mathbf{M}(k)$ is incrementally generated as follows:

$$\mathbf{M}(k) = \mathbf{M}(k-1) \oplus \mathbf{T}(k) \tag{5}$$

where "$\oplus$" is the XOR operator and $\mathbf{T}(k)$ is an intermediate binary template randomly created with $\rho \times l$ ones for the environmental period k. With this DOP generator, the random environment can be constructed. The parameter ρ controls the severity of the environmental changes while τ controls the change speed. It is worth noting that the environment changes at every τ fitness evaluations in this paper. This is different from Yang and Yao (2005, 2008), where the environment changes every τ generations.

With the DOP generator, cyclic dynamic environments are constructed as follows. First, we can generate $2K$ XORing masks as the base states in the search space randomly (see Yang 2005a; Yang and Yao 2005 for details). Then, the environment can cycle among these base states in a fixed logical ring. Furthermore, for constructing cyclic with noise environment, each time the XORing mask $\mathbf{M}(k)$ moves to the next state after bitwise flipping with a small probability p_n, called the noise rate in this paper. For the first period $k = 1$, $\mathbf{M}(1)$ is set to a zero vector. Then, the population at generation t is evaluated as follows:

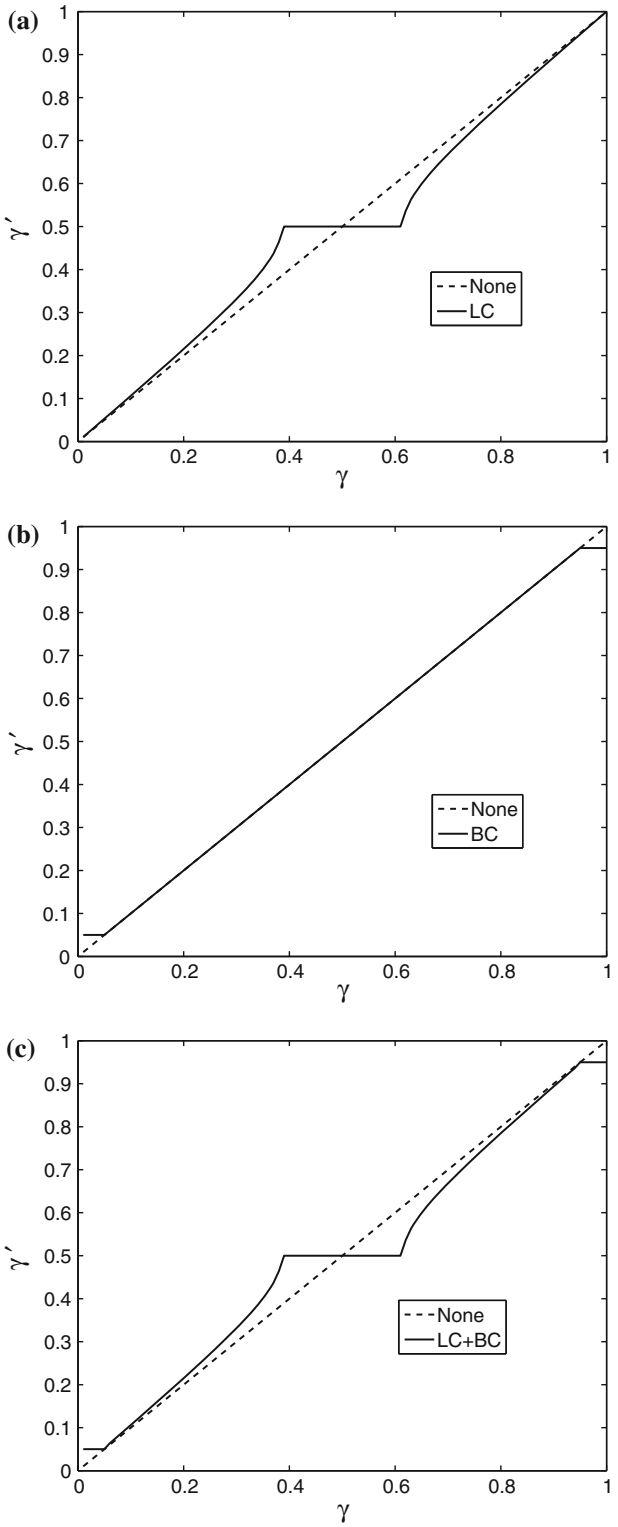

Fig. 5 The effect of LC, BC, and LC + BC on γ', assuming $N_{pop} = 20$, $\beta = 0.05$, and $p_s = 0.5$. **a** LC, **b** BC, **c** LC + BC

$$f(\mathbf{x}, t_e) = f(\mathbf{x} \oplus \mathbf{M}(k)) \tag{6}$$

where t_e is the fitness evaluation number and $k = \lceil t_e/\tau \rceil$ is the environmental period index.

In order to measure the performance of algorithms, the collective mean fitness (Morrison and De Jong 1999) is used in this paper. This measurement calculates the average of the best-of-generation fitness across the whole generations. Suppose each experiment is performed N_E times independently with the same experimental settings, the collective mean fitness (F_{CMF}) is formulated as follows:

$$F_{CMF} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N_E}\sum_{j=1}^{N_E} F_{BOG}(i,j)\right) \tag{7}$$

where G is the total generation number and N_e denotes the quantity of the environment periods in each run and $F_{BOG}(i,j)$ denotes the best-of-generation fitness of the ith generation in the jth run.

In order to understand the effect of memory scheme and diversity compensation measures on the population diversity during the running of an algorithm, we also recorded the diversity of the population every generation. The diversity of the population at time t in the kth run of an algorithm on a DOP is defined as

$$\mathrm{Div}(k,t) = \frac{1}{l \times N_{pop}(N_{pop}-1)}\sum_{i=1}^{N_{pop}}\sum_{j\neq i}^{N_{pop}} \mathrm{HD}(i,j) \tag{8}$$

where l is the encoding length, N_{pop} is the population size, and $\mathrm{HD}(i,j)$ is the Hamming distance between the ith and jth individual in the population. The mean population diversity of an algorithm on a DOP at time t over N_E runs is calculated as follows:

$$\overline{\mathrm{Div}}(k,t) = \frac{1}{N_E}\sum_{k=1}^{N_E}\mathrm{Div}(k,t) \tag{9}$$

4.2 Test functions

Decomposable unitation-based functions (DUFs), such as trap and deceptive functions, have been widely studied in the EA community in the attempt to understand what constructs difficult problems for EAs, especially for GAs (Goldberg 2002). In this paper, in order to analyze the performance of investigated algorithms in dynamic environments, three DUFs (denoted DUF1, DUF2, and DUF3) are selected as the stationary test functions. Each DUF consists of 25 copies of 4-bit building blocks and each building block contributes a maximum value of 4 to the total fitness, as shown in Fig. 6. The building block of the three DUFs are defined in Eqs. (10), (11), and (12), respectively.

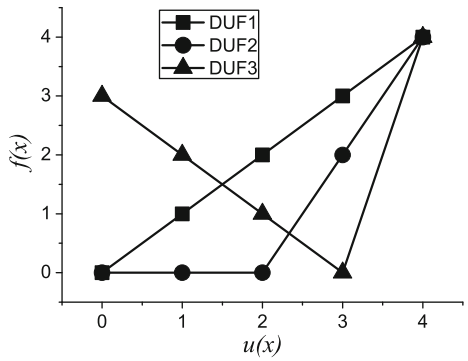

Fig. 6 The building block of the three DUFs

$$f_{DUF1}(x) = u(x) \tag{10}$$

$$f_{DUF2}(x) = \begin{cases} 4, & \text{if } u(x) = 4, \\ 2, & \text{if } u(x) = 3, \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$f_{DUF3}(x) = \begin{cases} 3 - u(x), & \text{if } u(x) < 4, \\ 4, & \text{otherwise} \end{cases} \tag{12}$$

where $u(x)$ denotes the number of ones in a building block.

DUF1 is, in fact, the OneMax function, which aims to maximize the number of ones in a chromosome. OneMax functions are usually taken as easy functions for EAs. For DUF2, in the search space of the 4-bit building block, the unique optimal solution is surrounded by only four sub-optimal solutions, while all the other 11 solutions form a wide plateau with zero fitness. The existence of this wide gap makes it much more difficultly for EAs to search on DUF2 than on DUF1. DUF3 is a fully deceptive function (Goldberg 2002). Fully deceptive functions are usually considered hard problems for EAs because the low-order building blocks inside the functions do not combine to form the higher order optimal building block: instead they combine into deceptive sub-optimal building blocks (Whitley 1991). Generally, the three DUFs form an increasing difficulty for EAs in the order from DUF1 to DUF2 to DUF3.

In this paper, the dynamic test problems are constructed by applying the XOR DOP generator to the three DUFs and the corresponding dynamic DUFs are denoted DDUF1, DDUF2, and DDUF3, respectively.

4.3 Experimental results and analysis

In this section, we present the results of four groups of experiments. The first group of experiments shows the validity of EI-MEDA by comparing it with EDAs without the environment identification based memory method. The second group of experiments compares the performance of EI-MEDA with some state-of-the-art algorithms for DOPs.

Table 1 The F_{CMF} value of UMDA, UMDA(LC + BC), EI-MUMDA, RUMDA, and RUMDA(LC + BC) over 50 runs on DDUFs in three types of dynamic environments

DDUF	Environmental type	UMDA	UMDA(LC + BC)	EI-MUMDA	RUMDA	RUMDA(LC + BC)
DDUF1	Cyclic	72.52	92.92	98.25	87.59	86.16
	Cyclic with noise	57.93	92.46	95.25	87.62	86.15
	Random	50.64	89.20	89.23	87.60	86.13
DDUF2	Cyclic	52.14	86.01	96.39	73.97	71.02
	Cyclic with noise	29.09	85.07	89.74	73.90	70.95
	Random	19.24	76.67	76.91	73.95	71.00
DDUF3	Cyclic	51.45	69.37	77.09	54.78	53.32
	Cyclic with noise	38.06	70.82	72.75	54.70	53.34
	Random	33.04	66.02	66.18	54.76	53.38

The third group of experiments aims to show that EI-MEDA can fit for any binary-coded EDAs. Finally, in order to deeply understand the proposed EI-MEDA, the sensitivity analysis of the effect of key parameters in EI-MEDA is also carried out in the fourth group of experiments.

In the experiments, some common settings are given as follows. Each algorithm was run 50 times in each experiment (i.e., $N_E = 50$). The total number of environmental changes N_e was set to 200. The dimension of each DDUF is 100 (i.e., each DDUF is encoded with 100 bit binary strings). The memory size m was set to 20 and the truncation selection rate p_s was set to 0.5.

4.3.1 Validation of EI-MEDA

The purpose of this group of experiments is to verify the validity of EI-MEDA. In EI-MEDA, the LC + BC and EI-MMS schemes work together to maintain the population diversity. The former works when the algorithm searches in a static environment while the latter works to respond to an environmental change. From this point of view, we compare EI-MEDA with several variants of EDAs, using the univariate marginal distribution algorithm (UMDA) (Mühlenbein and Paaß 1996) as an example EDA. We compare the following four algorithms: the original UMDA (denoted by UMDA), UMDA with LC + BC [denoted by UMDA(LC + BC)], UMDA with both EI-MMS and LC + BC (i.e., EI-MUMDA), UMDA with restart method (denoted by RUMDA), and UMDA with restart and LC + BC methods [denoted by RUMDA(LC + BC)]. The parameters are set as follows: $\tau = 1{,}000, \rho = 0.2, p_n = 0.01$, and $N_{pop} = 100$.

Table 1 shows the performance of each algorithm. From Table 1, it can be seen that by introducing LC + BC into the conventional UMDA, the performance of the algorithm UMDA(LC + BC) is enhanced a lot since the population diversity loss is compensated in each generation. In addition, if the EI-MMS scheme which reacts to environmental changes is applied to reuse memory information, the performance of the algorithm can be further enhanced. Therefore, it is obvious that no matter how the environment changes, EI-MUMDA benefits from both LC + BC and EI-MMS and performs the best on all DDUFs.

In contrast to EI-MUMDA, the algorithms that use the restart method [RUMDA and RUMDA(LC + BC)] compensate the population diversity in a totally blind way and hence, their performance is worse. Since RUMDA and RUMDA(LC + BC) do not use any historic information and restart from scratch when the environment changes, their performance is not greatly affected by the environmental dynamics type (i.e., cyclic, cyclic with noise, or random).

Another noticeable result is that although both the restart and LC + BC schemes improve the performance of UMDA [i.e., both RUMDA and UMDA(LC + BC) beats UMDA], it is not good to use them together in UMDA [i.e., the performance of RUMDA(LC + BC) is worse than both RUMDA and UMDA(LC + BC)]. This happens because the effect of enhancing the diversity level by the restart and LC + BC schemes may be too strong for RUMDA(LC + BC) to perform efficient search in a new environment.

Figure 7 shows the average dynamic population diversity of four algorithms in the first 100×100 fitness evaluations (i.e. ten environmental changes). From Fig. 7, it can be seen that UMDA poorly maintains its population diversity and can not adapt for the environmental changes. UMDA(LC + BC) can maintain its population diversity at a minimal diversity level using the LC + BC method. In each generation, at least $1/l \times N_{pop}$ individuals are randomly generated by the LC + BC method. When the environment changes, the population has to converge to the new optimum using the learning and sampling operations of the conventional UMDA. This leads to the small fluctuations in the population diversity level of UMDA(LC + BC).

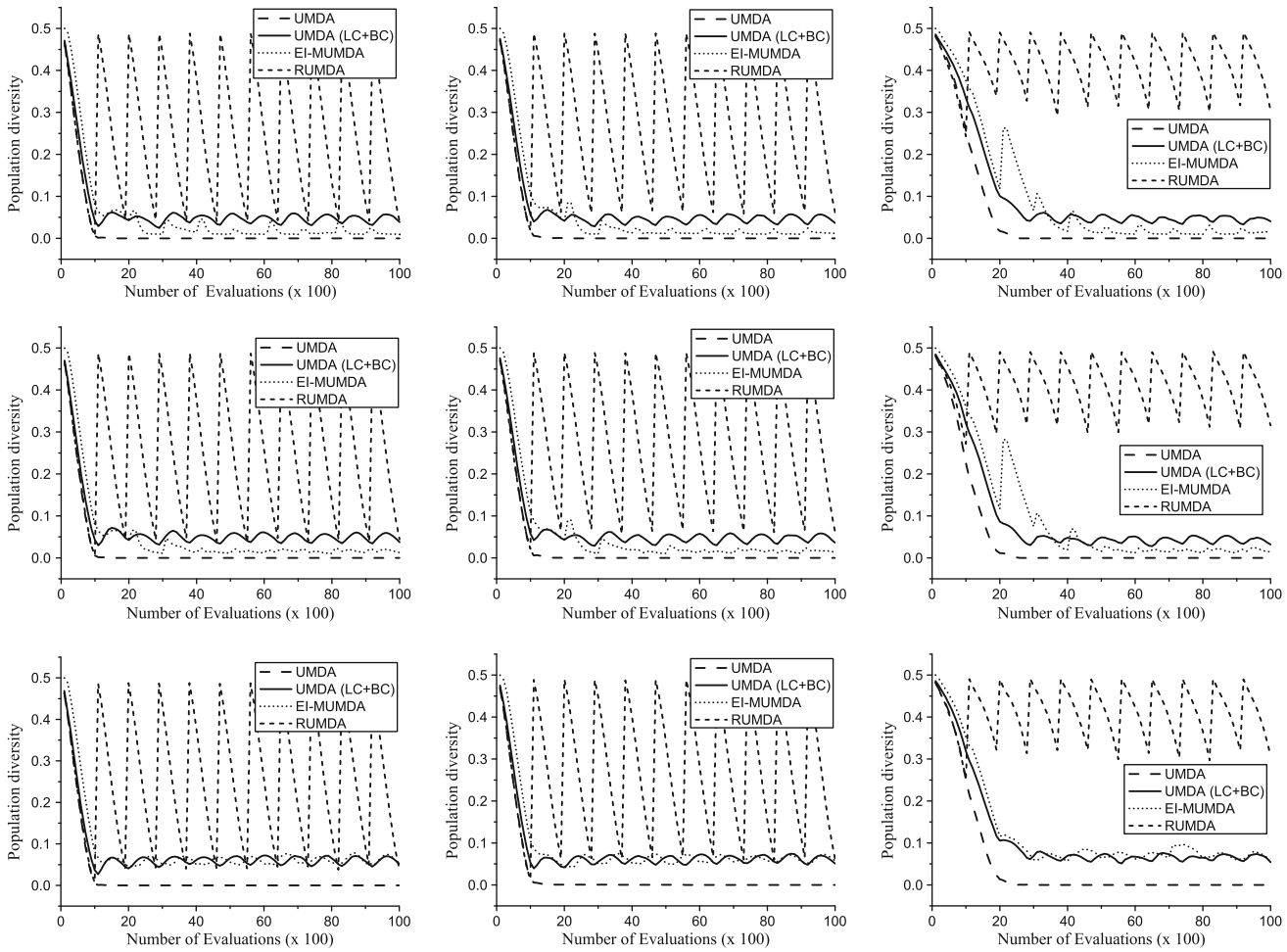

Fig. 7 The average dynamic population diversity of algorithms over 50 runs in the first 100×100 fitness evaluations on DDUF1 (*left column*), DDUF2 (*middle column*), and DDUF3 (*right column*) in three types of environments: cyclic environment (*top row*), cyclic environment with noise (*middle row*), and random environment (*bottom row*)

The LC + BC method is originally designed for compensating diversity in static environment and is lack of efficiency to track the moving optimum. If the information in the past optimization process could be used to infer the distribution of the new optimum and the population could be heuristically generated, the efficiency of dynamic optimization would be enhanced greatly.

According to this idea, EI-MUMDA stores the past probability models and reuse them to generate the initial population in a new environment. In the environment with cyclic or cyclic with noise environments, a probability model in the memory will be refined if it is retrieved back. In addition, by using the environment identification, the initial population in an environment can be generated around the possible optimum according to the memory. This is more efficient than the conventional evolutionary operators.

From the first two rows in Fig. 7, it can be seen that the diversity level of EI-MUMDA is lower and more smooth than that of UMDA(LC + BC). In the randomly changing environment, from the bottom row of Fig. 7, it can be seen that the diversity level curves of EI-MUMDA and UMDA(LC + BC) overlap with each other. This is because the memory in EI-MUMDA can not refine its elements (probability models) properly due to the randomly changing environment. Therefore, one can say that EI-MEDA performs well in dynamic environments, especially in dynamic environments with cyclic characteristic. As for RUMDA and RUMDA(LC + BC), although the restart method enables the original UMDA to react to environmental changes, these two algorithms are still defeated by EI-MUMDA. Because the restart method compensates population diversity in a blind and random way. As shown in Fig. 7, when environment changes (at every 1,000 fitness evaluations), the population is regenerated randomly and the diversity level goes up to about 0.5. In such a blind diversity compensation way, no useful information can be used to guide the population to track the optimum.

4.3.2 Comparison with the state-of-the-art algorithms

In this group of experiments, we compare EI-MUMDA with the following three algorithms: memory enhanced population-based incremental learning algorithm (MPBIL) (Yang 2005b), MPBIL with two populations and restart scheme (MPBIL2r) (Yang and Yao 2008), and random immigrants GA (RIGA) (Grefenstette and Fitzpatrick 1992). For MPBIL, an explicit memory is applied, which is randomly initialized and regularly updated. For MPBIL2r, a second population with restart method is added based on MPBIL. The population sizes of the two populations in MPBIL2r are adjustable according to their performance. When the environment changes, the first population searches associated with the memory while the second population searches from scratch. For RIGA, it differs from standard GA only in that in each generation, a set of worst individuals in the population are replaced by random immigrants. In the following experiments, the learning rate and memory size for MPBIL and MPBIL2r were set to 0.25 and 20, respectively. The crossover probability, mutation probability, and immigrant rate for RIGA were set to 0.6, 0.1, and 0.1, respectively. The population size of each algorithm was set to 100.

Figure 8 plots the performance (F_{CMF}) of each algorithm over 50 runs on different DDUFs, and the Wilcoxon rank sum test results of comparing EI-MUMDA with MPBIL, MPBIL2r, and RIGA are presented in Table 2, where "+", "−", or "∼" mean that the first algorithm is significantly better than, significantly worse than, or statistically equivalent to the second algorithm, respectively. The sample size and significant level of the Wilcoxon rank sum test are 50 and 0.05, respectively.

From the experimental results in Fig. 8 and Table 2, it can be seen that EI-MUMDA performs significantly better than RIGA and defeats MPBIL and MPBIL2r in most situations. The main reason lies in that EI-MUMDA updates its memory when an environmental change takes place. In this way, the probability models stored in the memory are improved as far as possible during two environmental changes. High-quality probability models can characterize the environments better and more likely represent the probability distribution of the optimum. As a result, saving and retrieving these high-quality probability models enable algorithms to track the optimum better.

4.3.3 Testing the effect of EI-MMS for binary-coded EDAs

In order to verify that the EI-MMS scheme can work effectively for binary-coded EDAs, we apply EI-MMS to the binary-coded UMDA and Bayesian optimization algorithm (BOA) (Pelikan 2002), respectively. The former is the simplest EDA, where the decision variables are independent to each other. In contrast, the latter is a complex EDA, where the relationships between the decision variables are modelled by a Bayesian network. The corresponding algorithms are denoted by EI-MUMDA and EI-MBOA respectively. Here, EI-MUMDA is compared with restart UMDA (RUMDA) and EI-MOBA is compared with restart BOA (RBOA) on DDUF2. In all of the above four algorithms, the LC + BC scheme is also used to compensate the diversity loss in the population. The relevant parameters were set as follows: the population size $N_{\mathrm{pop}} = 100, \tau = 1{,}000, \rho = 0.2$, and $p_{\mathrm{n}} = 0.01$.

Figure 9 shows the best fitness obtained by each algorithm in the first 50 environments. It can be seen that EI-MUMDA and EI-MBOA outperform RUMDA and RBOA respectively in all situations. This means that EI-MMS works effectively for both simple and complex binary-coded EDAs.

In addition, from Fig. 9 it can be seen that the variation of the performance of EI-MMS enhanced algorithms (EI-MUMDA and EI-MBOA) in each environment is affected by the environment type. The performance variation is small when the environment changes cyclically. When noise is added into the cyclic environment, the performance variation goes larger. The performance changes violently, e.g., the performance of restart algorithms (RUMDA and RBOA), in the random environment. The reason is that in a cyclic environment EI-MMS can perform very well to guide the EDA and reduce the blindness when a new environment comes. This is good for reducing the performance variation. When noise is added into the cyclic environment or the environment changes randomly, it becomes more difficult to correctly retrieve and update the memory. The inaccurate memory management increases the variation of the performance of EI-MUMDA and EI-MBOA.

4.3.4 Sensitivity analysis on the effect of parameters

In order to further understand EI-MEDA, in this group of experiments, we perform the sensitivity analysis of the effect of key parameters, including the environmental dynamics parameters, population size, and memory size, on the performance of EI-MEDA in dynamic environments. EI-MUMDA and RUMDA were used as example EDAs and, in order to draw some fair and general conclusions, the DDUF1 function was used as the test function in this group of experiments.

4.3.4.1 Effect of the environmental change speed First, we investigate how the environmental parameter τ affects the performance of EI-MEDA in the environments with different values of ρ. Table 3 presents the comparison between EI-MUMDA and RUMDA. Each element in the

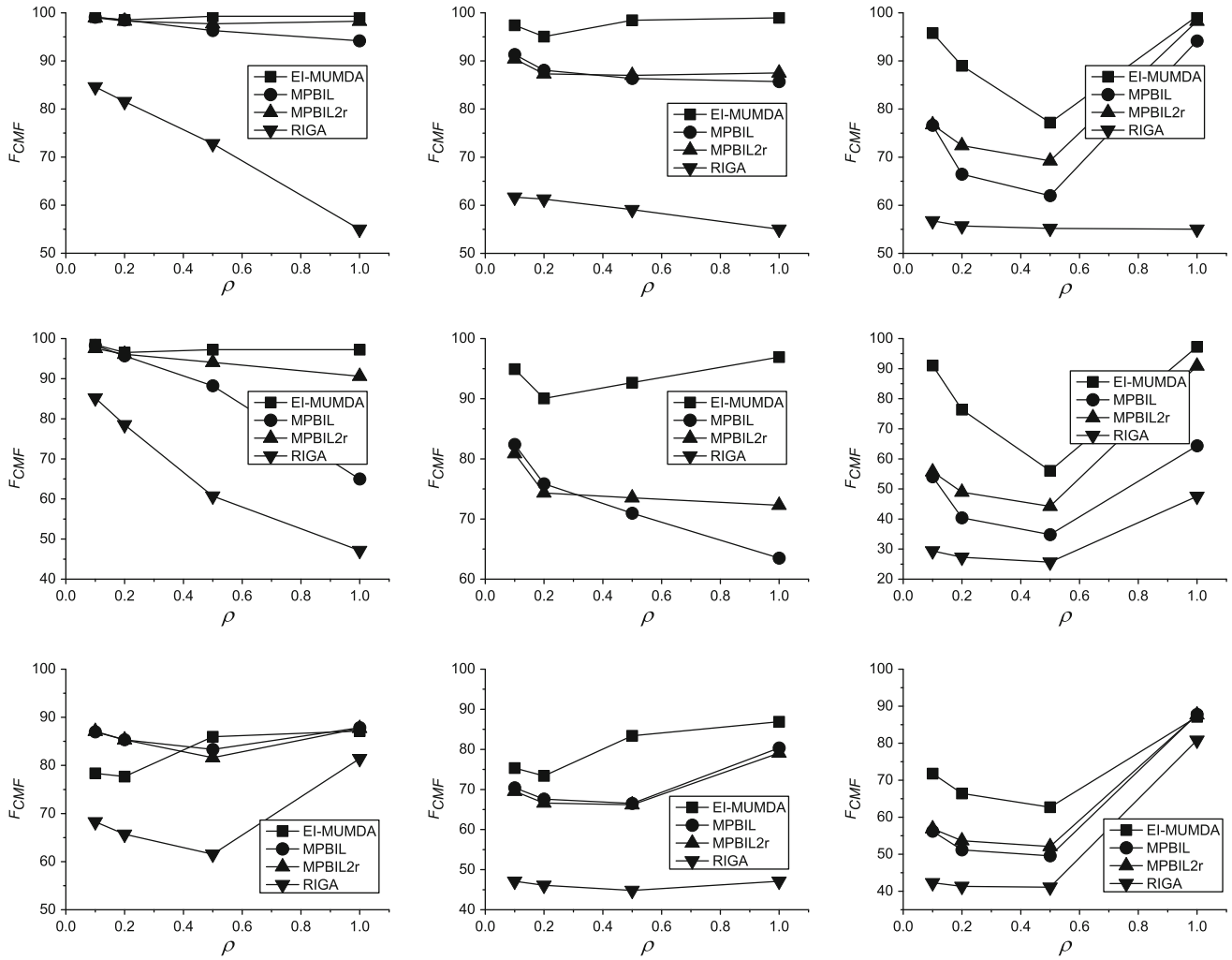

Fig. 8 The F_{CMF} values of EI-MUMDA, RIGA, MPBIL and MPBIL2r over 50 runs on DDUF1 (*left column*), DDUF2 (*middle column*), and DDUF3 (*right column*) in three types of environments: cyclic environment (*top row*), cyclic environment with noise (*middle row*), and random environment (*bottom row*)

table is the average performance difference between EI-MUMDA and RUMDA, i.e., $F_{\mathrm{CMF}}(EI{-}MUMDA) - F_{\mathrm{CMF}}(RUMDA)$, over 50 runs. Figure 10 shows the F_{CMF} value of EI-MUMDA under the environments with different τ and ρ. In the experiments, the population size N_{pop} was set to 100 and the noise rate p_{n} was set to 0.01.

From Fig. 10, it can be seen that the slower the environment changes (i.e., the larger the value of τ), the better EI-MUMDA can track the optimum dynamically. This is because a slowly changing environment involves a long static period between every two changes. Hence, EI-MUMDA can perform a better search during the static period and a memory element can be correctly selected according to the new environment.

As shown in Table 3, in the environments with cyclic characteristic (i.e., cyclic and cyclic with noise), the advantage of EI-MUMDA is more significant while the value of τ decreases. This means that EI-MMS enhances the algorithm to track the optimum more effectively in comparison with the restart scheme, especially in fast changing environments. For random environments, EI-MUMDA still outperforms RUMDA in general but is defeated when ρ is 0.5. The reason lies in that the environment with $\rho = 0.5$ is the most difficult to identify. If the algorithm can not correctly select a suitable memory element to retrieve, the new population sampled from it may miss the possible optimum.

In other words, the inaccurate environment identification may misguide the search in the new environment. This can also be demonstrated by Fig. 10c where EI-MUMDA performs the worst when the environmental dynamics parameter ρ is 0.5. When ρ is less or more than 0.5, the difficulty for environment identification is less. Extremely, when the environment changes completely every time, i.e., $\rho = 1.0$, the algorithm works well in a cyclic environment of two complementary states.

Table 2 The Wilcoxon rank sum test results of comparing EI-MUMDA with MPBIL, MPBIL2r, and RIGA on DDUFs in different environments, where " + " means significantly better, " − " means significantly worse, and "∼" means statistically equivalent

	ρ											
	DDUF1				DDUF2				DDUF3			
	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
Cyclic environment												
EI-MUMDA versus MPBIL	∼	∼	+	+	+	+	+	+	+	+	+	+
EI-MUMDA versus MPBIL2r	∼	∼	+	+	+	+	+	+	+	+	+	+
EI-MUMDA versus RIGA	+	+	+	+	+	+	+	+	+	+	+	+
Cyclic environment with noise												
EI-MUMDA versus MPBIL	∼	∼	+	+	+	+	+	+	+	+	+	+
EI-MUMDA versus MPBIL2r	+	∼	+	+	+	+	+	+	+	+	+	+
EI-MUMDA versus RIGA	+	+	+	+	+	+	+	+	+	+	+	+
Random environment												
EI-MUMDA versus MPBIL	−	−	+	−	+	+	+	+	+	+	+	−
EI-MUMDA versus MPBIL2r	−	−	+	−	+	+	+	+	+	+	+	−
EI-MUMDA versus RIGA	+	+	+	+	+	+	+	+	+	+	+	+

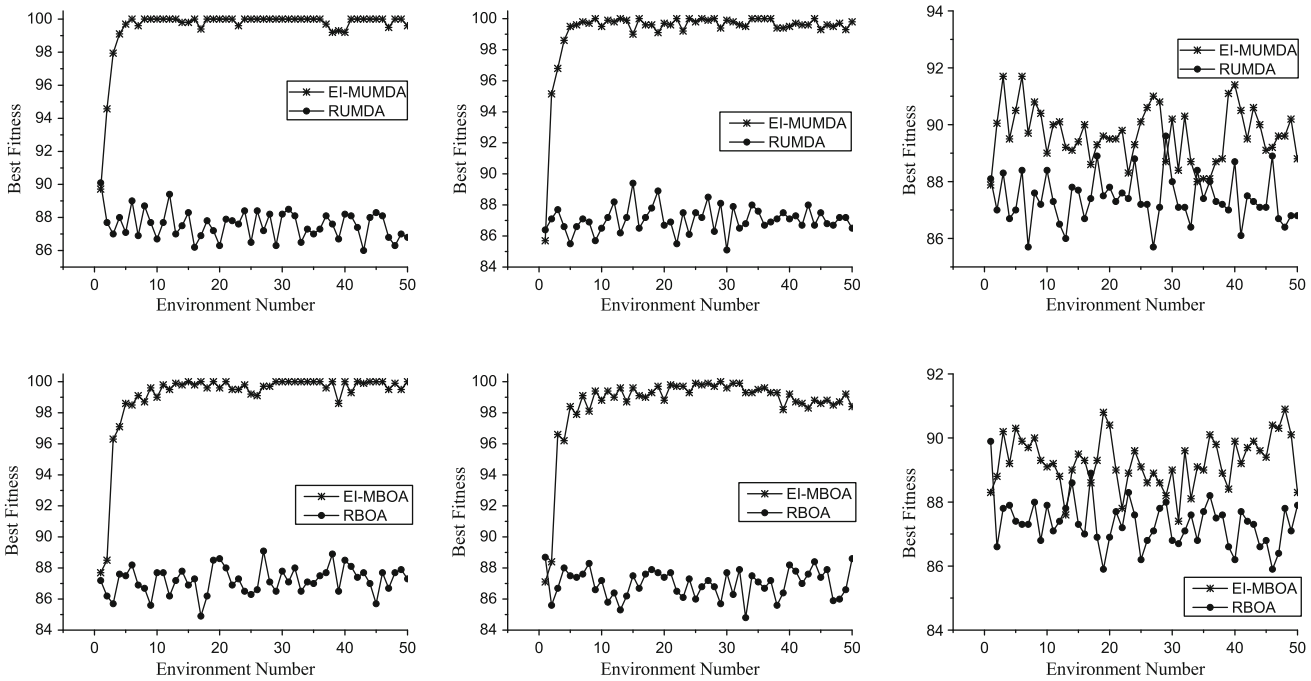

Fig. 9 The average best fitness of EI-MUMDA, EI-MBOA, RUMDA and RBOA in the first 50 environments over 50 runs on DDUF2 in cyclic environment (*left column*), cyclic environment with noise (*middle column*), and random environment (*right column*)

4.3.4.2 Effect of the noise rate p_n Second, we analyze how the noise rate p_n affects the performance of EI-MUMDA in dynamic environments. The parameters N_{pop} and τ were set to 100 and 1,000 respectively in the following experiments. Figure 11 shows $F_{CMF}(EI{-}MUMDA)$ in the dynamic environments with different p_n and ρ. Table 4 gives the comparison results between EI-MUMDA and RUMDA.

From Fig. 11, it can be observed that the performance of EI-MUMDA becomes better while the noise rate goes down. This reveals that noise is harmful for the algorithm. This is because noise makes the environment unable to return its previous base state exactly and hence no memory element can match a new environment exactly. In the XOR DOP generator (Yang 2005a; Yang and Yao 2008), for a cyclic with noise environment, before the problem moves

Table 3 The average difference of the F_{CMF} values between EI-MUMDA and RUMDA over 50 runs in three types of environments with different τ and ρ

	ρ			
	0.1	0.2	0.5	1.0
Cyclic environment				
$\tau = 200$	30.55	30.29	28.88	29.94
$\tau = 500$	23.26	22.53	22.86	22.98
$\tau = 1{,}000$	12.96	12.37	13.15	13.17
$\tau = 2{,}000$	16.15	5.67	6.21	6.27
Cyclic environment with noise				
$\tau = 200$	23.00	20.44	20.95	21.65
$\tau = 500$	20.61	17.99	20.86	22.07
$\tau = 1{,}000$	11.28	8.92	12.33	12.83
$\tau = 2{,}000$	5.21	3.71	5.99	6.09
Random environment				
$\tau = 200$	4.26	−0.06	−1.85	29.76
$\tau = 500$	12.73	2.59	−5.78	23.01
$\tau = 1{,}000$	9.67	2.86	−8.94	13.19
$\tau = 2{,}000$	4.73	1.66	−7.07	6.27

to a next environment, noise is added to an initial XORing mask that represents the base state of the new environment. This weakens the cyclic characteristic of the dynamic environment and hence is not good for the EI-MMS. Therefore, the larger the noise rate, the less likely that a memory element matches a new environment.

Observing the plots in Fig. 11 regarding $\rho = 0.1, 0.5$, and 1.0, EI-MUMDA performs better when the environmental change severity increases. The main reason is that the environmental change severity can offset the harmful effect caused by noise. Suppose the kth memory element is relevant to the new environment and its evaluation value should be the highest one. When ρ is large enough or p_{n} is relatively small, the fluctuation of the evaluation of the elements caused by the noise is slight in relative to the severity of environmental changes. Such slight fluctuation cannot affect the environment identification result much. This is demonstrated in Fig. 12a, where the current evaluation values (dashed line) of the elements are around the values according to the initial XORing mask (solid line). When ρ_1 is large enough in relative to p_{n}, the evaluation value of the proper element is still significantly higher than others and a correct environment identification can still be made.

However, when the environmental change severity is not large enough, the harmful influence caused by noise may not be effectively offset. In this condition, a larger environmental change severity makes the algorithm perform even worse. This is shown in Fig. 11 where the performance decreases when ρ changes from 0.1 to 0.2. This can be explained by Fig. 12b, when ρ_2 is not large enough, the proper element is confused with other elements due to the evaluation fluctuation. If the memory element can not be selected correctly, a larger environmental change severity means a larger gap between the optimum and the sampled population.

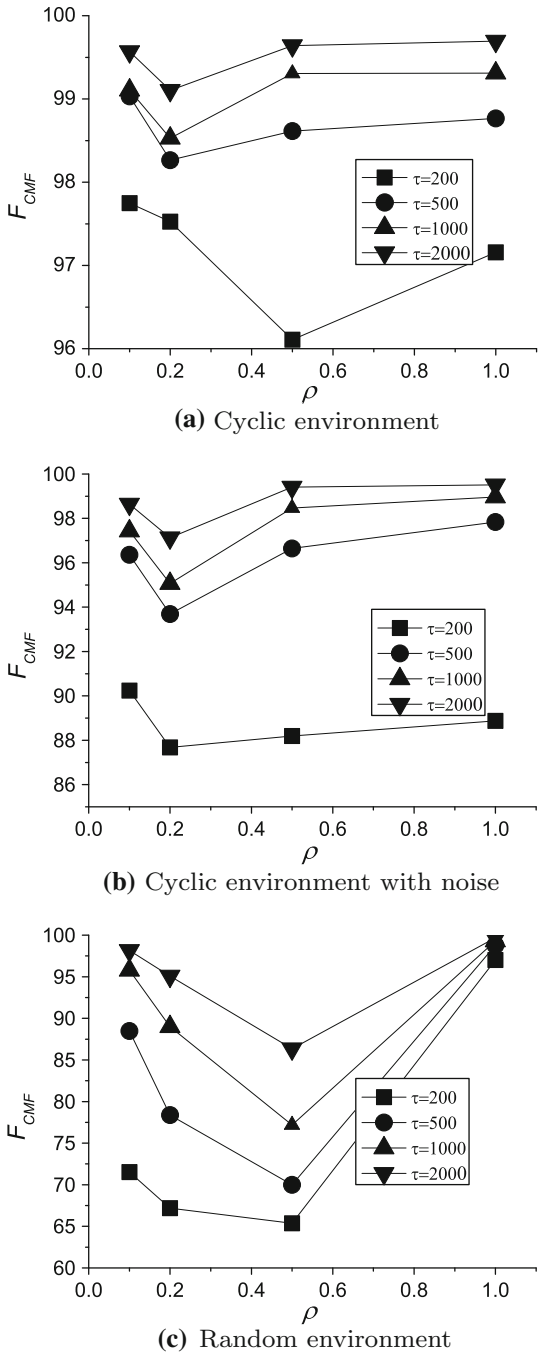

Fig. 10 The F_{CMF} value of EI-MUMDA over 50 runs in three types of environments with different τ and ρ

Besides, it can be seen from Table 4 that the advantage of EI-MUMDA over RUMDA decreases while the environmental noise rate rises. This reveals the fact that

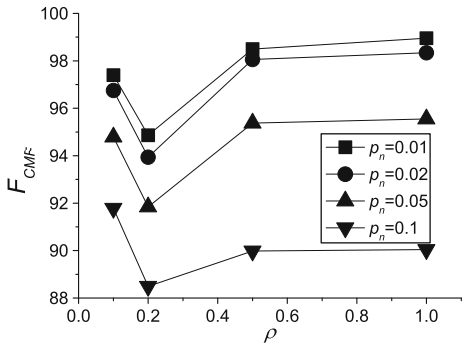

Fig. 11 The F_{CMF} value of EI-MUMDA over 50 runs in cyclic environments with noise with different p_{n} and ρ

Table 4 The average difference of F_{CMF} values between EI-MUMDA and RUMDA over 50 runs in cyclic environments with noise with different p_{n} and ρ

	ρ			
	0.1	0.2	0.5	1.0
$p_{\mathrm{n}} = 0.01$	11.25	8.71	12.32	12.80
$p_{\mathrm{n}} = 0.02$	10.59	7.77	11.92	12.19
$p_{\mathrm{n}} = 0.05$	8.64	5.70	9.21	9.43
$p_{\mathrm{n}} = 0.1$	5.65	2.34	3.83	3.89

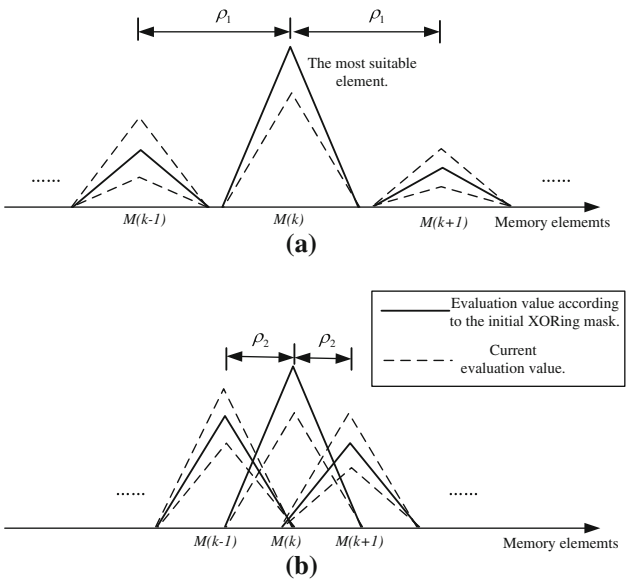

Fig. 12 Illustration of how ρ and p_{n} affect the environment identification: **a** ρ_1 is relatively large enough to offset the noise, **b** ρ_2 is not relatively large enough to offset the noise

EI-MUMDA is more sensitive to the environmental noise than RUMDA. Nevertheless, EI-MUMDA still outperforms RUMDA in all situations. That is, the EI-MMS still effectively enhances EI-MUMDA to react to environmental changes.

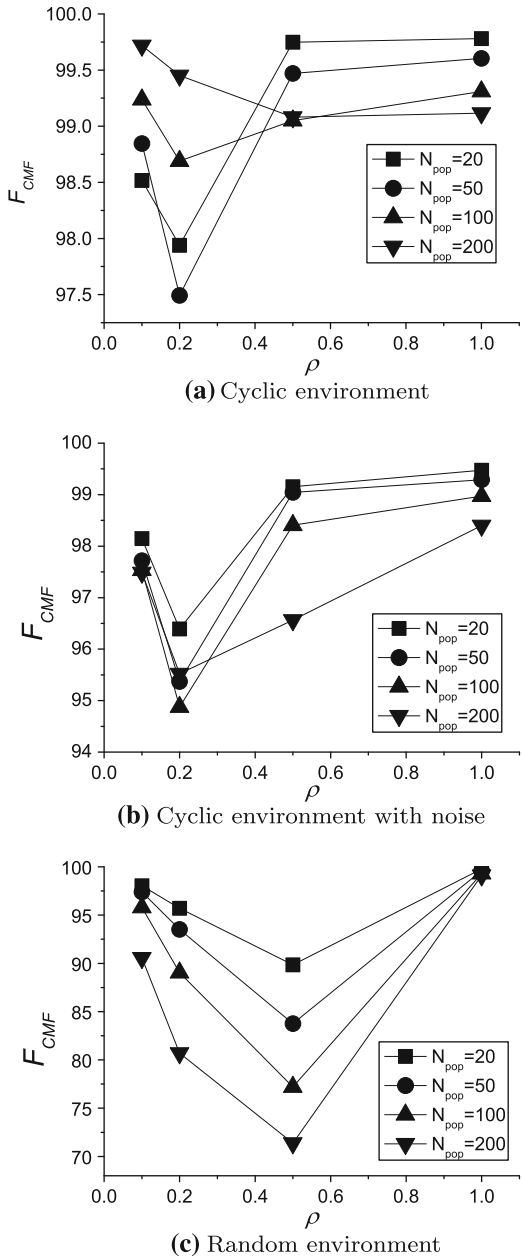

Fig. 13 The F_{CMF} value of EI-MUMDA over 50 runs in three types of environments with different N_{pop} and ρ

4.3.4.3 Effect of the population size This set of experiments was performed to analyze the sensitivity of the effect of the population size to the performance of EI-MUMDA. The environmental dynamics parameters were set as follows: $\tau = 1{,}000$ and $p_{\mathrm{n}} = 0.01$.

Figure 13 shows that in most situations EI-MUMDA performs worse when its population size increases. This is because a larger population size means more fitness evaluations in each generation. Therefore, within the same number of fitness evaluations, a smaller population will evolve more generations and the probability model will be

Table 5 The average difference of F_{CMF} values between EI-MUMDA and RUMDA over 50 runs in three types of environments with different N_{pop} and ρ

	ρ			
	0.1	0.2	0.5	1.0
Cyclic environment				
$N_{\mathrm{pop}} = 20$	5.15	4.57	6.34	6.42
$N_{\mathrm{pop}} = 50$	7.21	5.86	7.80	7.93
$N_{\mathrm{pop}} = 100$	13.08	12.52	12.91	13.14
$N_{\mathrm{pop}} = 200$	22.17	21.89	21.50	21.58
Cyclic environment with noise				
$N_{\mathrm{pop}} = 20$	4.79	3.01	5.77	6.10
$N_{\mathrm{pop}} = 50$	6.07	3.73	7.40	7.64
$N_{\mathrm{pop}} = 100$	11.40	8.69	12.28	12.84
$N_{\mathrm{pop}} = 200$	19.90	17.96	18.99	20.83
Random environment				
$N_{\mathrm{pop}} = 20$	4.67	2.32	− 3.53	6.39
$N_{\mathrm{pop}} = 50$	5.77	1.89	− 7.87	7.98
$N_{\mathrm{pop}} = 100$	9.63	2.89	− 8.94	13.16
$N_{\mathrm{pop}} = 200$	12.99	3.18	− 6.17	21.56

refined more times. This enables the EI-MMS to draw a better memory element from the corresponding environment and react to environmental changes better.

Table 5 shows the comparison between EI-MUMDA and RUMDA. It can be seen that, in most situations, the advantage of EI-MUMDA over RUMDA becomes more significant while the population size rises. This means that the heuristic effect of EI-MMS is important for the algorithm to react to the changing environment. When the computational burden for evolving the population is heavy, the EI-MMS is more effective than the restart method to help the algorithm track the changing optimum.

4.3.4.4 Effect of the memory size Finally, we investigate how the memory size affects the performance of EI-MEDA. The following experiments were carried out to test the performance of EI-MUMDA with different memory sizes $m \in \{5, 10, 20, 40\}$. Some other parameters were set as follows: $\tau = 1{,}000$, $\rho = 0.1$, $p_{\mathrm{n}} = 0.01$, and $N_{\mathrm{pop}} = 100$.

Figure 14 shows the performance of EI-MUMDA with different memory sizes in different environments on the three DDUFs. It can be seen that in cyclic environments, when the memory size $m \leq 20$, the performance of EI-MUMDA improves as the value of m increases. However, when $m > 20$ (i.e., $m = 40$), the performance of EI-MUMDA does not change significantly. This is because in this experiment, there are 20 (i.e., $2/\rho$) intermediate binary templates in the bitwise XOR DOP generator, which means the environment re-cycles after it changes 20 times.

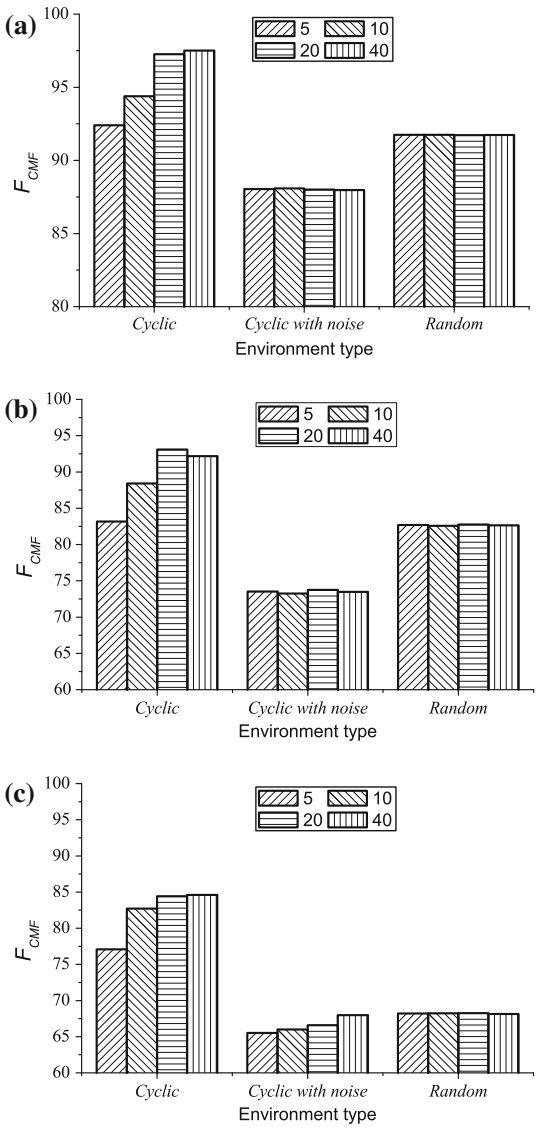

Fig. 14 The F_{CMF} value of EI-MUMDA with different memory sizes over 50 runs in different environments on **a** DDUF1, **b** DDUF2, and **c** DDUF3

Therefore, the algorithm needs at most 20 memory elements to store the probability models obtained in each environment. The redundant memory elements when $m = 40$ cannot significantly enhance the performance any more.

For the cyclic with noise environment, if p_{n} is large or the environmental change severity is small, the memory elements may be close to each other. As a result, the environment identification method cannot work properly. For example, from Fig. 14, it can be seen that when the environment is cyclic with noise, the memory size seems not a sensitive parameter to affect the performance of EI-MUMDA. In a similar way, a randomly changing environment may also weaken the positive effect of memory. Therefore, it can be seen from Fig. 14 that when the

environment changes randomly, the memory size does not affect the performance of EI-MUMDA significantly.

In summary, several conclusions can be drawn from the above experiments on the sensitivity analysis of parameters: (1) a slowly changing environment is good for EI-MEDA to track the moving optimum; (2) noise is a negative factor for EI-MEDA and the severity of the environmental changes can offset this factor to some degree. If noise can be effectively offset, a severely changing environment is good for the environment identification; otherwise, a large environmental change severity may make the situation worse. (3) A large population needs more computational effort to evolve and this degrades the performance of EI-MEDA. Nevertheless, the EI-MMS helps an EDA track dynamic optimum more effectively than the restart method. (4) If the environment is easy to identify and the memory size is smaller than $2/\rho$, a large memory size is positive to enhance the performance of EI-MEDA; otherwise, the memory size may not affect the performance of EI-MEDA significantly.

5 Conclusions

In this paper, an environment identification based memory management scheme (EI-MMS) is proposed to enhance the performance of binary-coded estimation of distribution algorithms (EDAs) for dynamic optimization problems (DOPs). In EDAs with the EI-MMS (i.e., EI-MEDA), probability models are taken as memory elements due to their ability to characterize each environment. When the environment changes, the probability model generated in the previous generation is stored. Then, a suitable element in the memory is used to generate the initial population that may be near the possible optimum in the new environment. In order to retrieve a suitable memory element which matches a new environment, an environment identification method which combines the sample averaging and best individual schemes is proposed in the EI-MMS. Since the diversity of conventional EDAs will loss gradually during the learning and sampling processes of the probability models, an effective diversity compensation method which combines the loss correction and boundary correction schemes is also introduced into EI-MEDA to further enhance its performance in dynamic environments.

In order to test the validity of the EI-MMS, several groups of experiments have been carried out based on three dynamic decomposable unitation-based functions (DDUFs) in three types of environments. The experimental results show that the EI-MMS is valid to improve the performance of EDAs for DOPs and that EI-MEDA is suitable for any binary EDAs to track moving optimum, especially in the environments with cyclic characteristics (i.e., cyclic environments and cyclic environments with noise). In the experiments, EI-MEDA is also applied to the univariate marginal distribution algorithm (UMDA) and the results show its advantage over other three peer algorithms, i.e., the memory enhanced population-based incremental learning algorithm (MPBIL) (Yang 2005a), MPBIL with two populations and restart scheme (MPBIL2r) (Yang and Yao 2008), and random immigrants GA (RIGA) (Grefenstette and Fitzpatrick 1992), on most cases. In order to understand the proposed method more deeply, the sensitivity analysis on how the key parameters (such as the environmental change speed and severity, the noise rate in cyclic environments, the population size, and the memory size) affect the performance of EI-MEDA has also been carried out in this paper.

Generally, the experimental results indicate that the proposed EI-MMS is efficient in enhancing the performance of EDAs for DOPs and the corresponding EI-MEDAs are good choices for DOPs.

Acknowledgments The authors would like to thank the anonymous associate editor and reviewers for their thoughtful suggestions and constructive comments. This work was supported by the National Nature Science Foundation of China (NSFC) under Grant 60774064, the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/01.

References

Branke J (1999) Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 IEEE congress on evolutionary computation (CEC99), vol 3, pp 1875–1882

Branke J (2001) Evolutionary optimization in dynamic environments, Kluwer, Dordrecht

Branke J, Kaubler T, Schmidt C, Schmeck H (2000) A multipopulation approach to dynamic optimization problems. In: Proceedings of the 4th international conference on adaptive computing in design and manufacturing, pp 299–308

Branke J, Lode C, Shapiro JL (2007) Addressing sampling errors and diversity loss in UMDA. In: Proceedings of the 9th annual conference on genetic and evolutionary computation (GECCO 2007), pp 508–515

Cobb HG (1990) An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Techincal Report AIC-90-001, Naval Research Lab, Washington, USA

Cedeno W, Vemuri VR (1997) On the use of niching for dynamic landscapes. In: Proceedings of the 1997 IEEE international conference on evolutionary computation, pp 361–366

Goldberg DE (2002) The design of innovation: lessons from and for competent genetic algorithms. Kluwer, Norwell

Grefenstette JJ, Fitzpatrick J (1992) Genetic algorithms for changing environments. In: Proceedings of the 2nd internatioanl conference on parallel problem solving from nature

Jin Y, Branke J (2005) Evolutionary optimization in uncertain environments—a survey. IEEE Trans Evol Comput 9(3): 303–317

Mori N, Kita H, Nishikawa Y (1996). Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In: Ebeling W et al. (eds) Proceedings of the 4th international conference on parallel problem solving from nature (PPSN IV), pp 513–522

Morrison R (2004) Designing evolutionary algorithms for dynamic environments, Springer, Berlin

Morrison R, De Jong K (1999) A test problem generator for non-stationary environments. In: Proceedings of the 1999 congress on evolutionary computation, pp 2047–2053

Mühlenbein H, Paaß G (1996) Recombination of genes to the estimation of distributions. In: Ebeling W (ed) Proceedings of the 4th international conference on parallel problem solving from nature (PPSN IV), pp 178–187

Pelikan M (2002) Bayesian optimization algorithm: from single level to hierarchy. PhD thesis. University of Illinois, Urbana-Champaign, USA

Shapiro JL (2003) Scaling of probability-based optimization algorithms. In: Obermayer K (ed) Advances in neural information processing systems, pp 399–406

Shapiro JL (2005) Drift and scaling in estimation of distribution algorithms. Evol Comput 13(1):99–123

Shapiro JL (2006) Diversity loss in general estimation of distribution algorithms. In: Runarsson TP (ed) Proceedings of the 9th international conference on parallel problem solving from nature, LNCS 4193, pp 92–101

Ursem RK (2000) Multinational GA optimization techniques in dynamic environments. In: Proceedings of the 2000 genetic and evolutionary computation conference (GECCO 2000), pp 19–26

Whitley LD (1991) Fundamental principles of deception in genetic search. In: Rawlins GJE (ed) Foundations of genetic algorithms. Morgan Kaufmann, San Francisco, pp 221–241

Wineberg M, Oppacher F (2000) Enhancing the GA's ability to cope with dynamic environments. In: Proceedings of the 2000 Genetic and Evolitionary Computation Conference (GECCO 2000), pp 3–10

Yang S (2003) Non-stationary problem optimization using the primal-dual genetic algorithm. In: Proceedings of the 2003 IEEE congress on evolutionary computation, vol 3, pp 2246–2253

Yang S (2005a) Memory-enhanced univariate marginal distribution algorithms for dynamic optimization problems. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol 3, pp 2560–2567

Yang S (2005b) Population-based incremental learning with memory scheme for changing environments. In: Proceedings of the 7th annual conference on genetic and evolutionary computation (GECCO 2005), pp 711–718

Yang S (2006) Associative memory scheme for genetic algorithms in dynamic environments. In: EvoWorkshops 2006: applications of evolutionary computing, pp 788–799

Yang S, Yao X (2005) Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput 9(11):815–834

Yang S, Yao X (2008) Population-based incremental learning with associative memory for dynamic environments. IEEE Trans Evol Comput 12(5):542–561