Adaptive Primal–Dual Genetic Algorithms in Dynamic Environments

Hongfeng Wang, Shengxiang Yang, *Member, IEEE*, W. H. Ip, and Dingwei Wang

***Abstract*—Recently, there has been an increasing interest in applying genetic algorithms (GAs) in dynamic environments. Inspired by the complementary and dominance mechanisms in nature, a primal–dual GA (PDGA) has been proposed for dynamic optimization problems (DOPs). In this paper, an important operator in PDGA, i.e., the primal–dual mapping (PDM) scheme, is further investigated to improve the robustness and adaptability of PDGA in dynamic environments. In the improved scheme, two different probability-based PDM operators, where the mapping probability of each allele in the chromosome string is calculated through the statistical information of the distribution of alleles in the corresponding gene locus over the population, are effectively combined according to an adaptive Lamarckian learning mechanism. In addition, an adaptive dominant replacement scheme, which can probabilistically accept inferior chromosomes, is also introduced into the proposed algorithm to enhance the diversity level of the population. Experimental results on a series of dynamic problems generated from several stationary benchmark problems show that the proposed algorithm is a good optimizer for DOPs.**

***Index Terms*—Adaptive dominant replacement scheme, dynamic optimization problem (DOP), genetic algorithm (GA), Lamarckian learning, primal–dual mapping (PDM).**

I. Introduction

As a class of widely used optimization techniques, genetic algorithms (GAs) have extended their application areas from simple functions into more complicated optimization problems, such as multiobjective optimization problems [6], multimodal optimization problems [23], and so on [7], [10]. Recently, studying GAs for dynamic optimization problems (DOPs) has attracted a growing interest from the GA community considering that many real-world optimization problems are often subject to dynamic environments [8], [24]. In dynamic environments, the fitness landscape may change over time as a result of the changes of optimization goal, problem instance, and/or some restrictions. For example, manufacturing businesses in today's markets are facing immense pressures to rapidly react to dynamic variations in demand distributions across products and changing product mixes [37]. For these DOPs, the goal of GAs is no longer to find a satisfactory solution to a fixed problem, but to track the trajectory of moving optima in the search space [2], [36]. This poses great challenges to traditional GAs because they cannot track the changing optimal solutions well once converged.

Although traditional GAs cannot address DOPs well, GAs, with proper enhancements, are good choices to solve DOPs. This is because GAs are inspired by natural and biological evolution, which is always subject to dynamic environments, and hence possess potential properties to adapt in dynamic environments. In recent years, several approaches have been developed into GAs to address DOPs [13]. These approaches can roughly be grouped into four categories: diversity schemes (increasing the population diversity after a change is detected [4], [26], or maintaining the population diversity during the run [11], [31], [32]), memory schemes [1], [28], [30], [36], multipopulation and speciation schemes [3], [19], [20], and adaptive schemes [4], [16], [33], [34].

Inspired by the complementary mechanism in nature, a variant of GA, called primal–dual GA (PDGA), was proposed by Yang [29] and shown to be suitable for addressing binary-encoded DOPs. In PDGA, each primal chromosome, which is explicitly recorded in the population, has its dual chromosome, which is calculated using a primal–dual mapping (PDM) operator. At each generation, a set of chromosomes in the population is selected to evaluate their duals before the next generation starts, and a dominant replacement scheme is used to decide whether the duals can replace the selected primal chromosomes. Recently, the original PDGA has been improved with some preliminary experimental results in [27].

In this paper, the PDM operator in PDGA is further investigated to improve its robustness and adaptability in dynamic environments. The PDM operator was originally designed as the maximum Hamming distance between a pair of primal–dual chromosomes. That is, each allele in the gene locus of the primal chromosome is translated to its complement during the mapping course (e.g., from 0 to 1 or from 1 to 0 if a binary-encoded optimization problem is assumed here). Instead of the original PDM operator, an adaptive PDM operator is proposed in this paper. It is a probability-based function where the translation on every bit in a chromosome string to its complement is executed according to a mapping probability that is adjusted using the statistical information of the distribution of alleles in a gene locus over the population. In addition to

Manuscript received July 16, 2008; revised November 7, 2008 and January 30, 2009. First published April 21, 2009; current version published November 18, 2009. This work was supported in part by the National Nature Science Foundation of China (NSFC) under Grant 70431003 and Grant 70671020, by the National Innovation Research Community Science Foundation of China under Grant 60521003, by the National Support Plan of China under Grant 2006BAH02A09, by the Engineering and Physical Sciences Research Council (EPSRC) of U.K. under Grant EP/E060722/1, and by the Hong Kong Polytechnic University Research Grants under Grant G-YH60. This paper was recommended by Associate Editor Y. S. Ong.

H. Wang and D. Wang are with the School of Information Science and Engineering, Northeastern University, Shenyang 110004, China (e-mail: hfwang@mail.neu.edu.cn; dwwang@mail.neu.edu.cn).

S. Yang is with the Department of Computer Science, University of Leicester, Leicester LE1 7RH, U.K. (e-mail: s.yang@mcs.le.ac.uk).

W. H. Ip is with the Department of Industrial and Systems Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: mfwhip@inet.polyu.edu.hk).

Digital Object Identifier 10.1109/TSMCB.2009.2015281

this adaptive PDM operator, an adaptive dominant replacement approach, where a chromosome can be replaced by its dual with a probability, is also proposed in this paper to increase the population diversity.

In this paper, extensive experiments are carried out to compare the performance of the proposed adaptive PDGA with several peer GAs based on a series of dynamic test environments, which are systematically constructed from several stationary functions using the DOP generator proposed in [29] and [35]. Based on the experimental results, an algorithm performance analysis with respect to the weakness and strength of the studied algorithms in dynamic environments is carried out. This paper also carries out experiments on the sensitivity analysis with respect to several important factors, such as the PDM operators and the replacement scheme, on the performance of the adaptive PDGA in dynamic environments.

The rest of this paper is organized as follows. Section II briefly reviews existing complementary approaches for GAs in dynamic environments. Section III details the proposed PDGA with the adaptive PDM scheme. Sections IV and V describe the dynamic test environments for this paper and present the experimental results and relevant analysis, respectively. The conclusions and discussions on future work are given in Section VI.

II. Relevant Work

For traditional GAs in dynamic environments, the main problem lies in that all chromosomes may eventually converge to an optimum point and hence loose their diversity. Therefore, traditional GAs cannot adapt well to a changing environment. To address the convergence problem, researchers have introduced the complementary mechanism in nature into GAs to improve their performance for DOPs. The relevant works are briefly reviewed below.

The most prominent complementary mechanism-based approaches are to use multiploidy chromosomes for the representation of individuals. Goldberg and Smith [9] first proposed a diploidy-based GA with a tri-allelic dominance scheme for the time-varying knapsack problem. Thereafter, Ng and Wong [17] investigated a dominance scheme with four possible alleles for a diploid GA and reported a better performance than the tri-allelic scheme. Hadad and Eick [12] used multiploidy and a dominance vector as an additional part of an individual that breaks the ties whenever there are an equal amount of 0's and 1's at a specific gene location. Ryan [21] used an additive multiploidy, where the genes determining one trait are added to determine the phenotypic trait. The phenotypic trait becomes 1 if a certain threshold is exceeded, or 0, otherwise. Lewis *et al.* [14] compared several multiploid approaches and observed some interesting results. For example, a simple dominance scheme is not sufficient to track the changing optimum well, but much better results can be obtained if the method is extended with a dominance change mechanism. Recently, Uyar and Harmanci [25] proposed an adaptive dominance change mechanism for diploid GAs, where the dominance characteristics for each locus are dynamically adjusted via the feedback from the current population.

```
Procedure general PDGA
begin
  parameterize(pop_size,pc,pm);
  t := 0;
  initializePopulation(P(0));
  evaluatePopulation(P(0));
  D(0):=selectForDualEvaluation(P(0));
  for each chromosome x in D(0) do
    executePDMOperation(x);
  endfor
  repeat
    P'(t) := selectForRecombination(P(t));
    P''(t) := crossover(P'(t));
    mutate(P''(t));
    evaluatePopulation(P''(t));
    P(t+1) :=selectForSurvial(P(t)+P''(t));
    D(t+1):=selectForDualEvaluation(P(t+1));
    for each chromosome x in D(t+1) do
      executePDMOperation(x);
    endfor
    t := t+1;
  until a termination condition is met;
end
```

Fig. 1. Pseudocode of the framework of PDGA.

Similar to multiploid approaches, the dualism mechanism was also introduced into GAs to improve their performance in dynamic environments. Collard *et al.* [5] proposed the dual GA (DGA), which was inspired by the principle of the dualism mechanism in nature. DGA operates on a search space by introducing a meta-gene in front of the regular bits. When the meta-gene is set to "0," it has no effect on the regular bits, but when the meta-gene is set to "1," all regular bits are translated to their compliment for fitness evaluation. Thus, each point in the search space has two complementary representations. For example, the two individuals [**0**011] and [**1**100] have the same phenotypic meaning. The added meta-gene bit undergoes the same genetic operations within DGA as other regular bits do.

III. Description of Algorithms Investigated

A. PDGA and the PDM Operator

Inspired by the dualism mechanism, Yang has proposed a PDGA for DOPs in the binary-encoded space [29]. Here, the framework of PDGA is first introduced, as simply shown in Fig. 1. Within PDGA, a population of pop_size chromosomes is randomly generated and evaluated at the initialization step. At each subsequent generation, the chromosomes are proportionally selected from the current population and undergo the one-point crossover operation with a probability pc. After crossover, the bit-wise mutation operation is performed for each newly generated offspring chromosome, which changes the allele value in each locus of an offspring bit wise (0 to 1 and vice versa) with a probability pm. Then, the pop_size best chromosomes among all parents and offspring are selected to proceed into the next generation, and a set $D(t)$ of lowest fitness chromosomes in the newly generated population is selected to perform PDM operations before the next generation starts.

It is obvious that a new genetic operator, i.e., the PDM operator, plays an important role during the running of PDGA.

Procedure general PDM operator

begin
x':=createDualChromosome(x);
evaluateChromosome(x');
if *a replacement condition is met* **then**
replace x in $P(t)$ with x';
end
Denotations:
$P(t)$: the population at generation t
x: the selected primal chromosome in $P(t)$
x': the generated dual chromosome after executing the PDM operation for x

Fig. 2. Pseudocode of a general PDM operator.

Primal chromosome:	1	1	0	0	1	0
PDM operation:	⇓	⇓	⇓	⇓	⇓	⇓
Dual chromosome:	0	0	1	1	0	1

Fig. 3. Example operation of the original PDM operator.

The PDM operation can be expressed by the pseudocode in Fig. 2. From Fig. 2, it can be observed that a PDM operation can include two steps: first, to create a chromosome's dual, which requires the definition of the PDM function between a pair of primal–dual chromosomes, and then to decide whether a primal chromosome should be replaced by its dual, which requires the design of a replacement scheme.

Within the original PDGA, the PDM function is defined as the maximum Hamming distance (the number of locations where the corresponding bits of two chromosomes differ) in the binary-encoded distance space. Given a primal chromosome $x = (x_1, x_2, \ldots, x_L) \in I = \{0, 1\}^L$ of fixed length L, its dual $x' = dual(x) = (x'_1, x'_2, \ldots, x'_L) \in I$, where $dual(\cdot)$ is a PDM function, and $x'_i = 1 - x_i$. Fig. 3 shows an example of applying the original PDM operator to a 6-bit string chromosome. For each individual $x \in D(t)$, if the fitness of its dual x' is calculated to be better, then the corresponding PDM operation is called *valid*, and x is replaced with x'; otherwise, x remains in the next generation if the PDM operation is invalid. Therefore, only a valid PDM operation gives the dual chromosome a chance to transfer into the next population, which is similar to the dominance mechanism in nature.

B. Probability-Based PDM Operator

The PDM operation can make a low-fitness chromosome quickly jump into the high-fitness area in the solution space. Thus, this operator can not only help accelerate the searching process to achieve satisfactory solutions but also enhance the PDGA's capability of adapting to a changing environment. The PDM function is originally designed as the maximum distance in the Hamming distance space. That is, each allele in the gene locus of a primal chromosome is translated to its complement during the PDM operation. It is obvious that the original PDM scheme can take effect particularly in the early searching stage of the algorithm or when the fitness landscape fluctuates with a sharp degree. However, this hypermapping scheme may become useless in several cases. For example, when most individuals have converged into the high-performance area in the late searching stage and hence their duals become inferior, or when the environment very slightly changes, the PDM operations become invalid.

1's frequency in a locus:	0.1	0.4	0.8	0.3	0.7	0.2
PDM probability:	0.74	0.26	0.58	0.42	0.42	0.58
Generate a random number:	0.7	0.3	0.6	0.5	0.4	0.2
Whether mapping:	Y	N	N	N	Y	Y
Primal chromosome:	1	1	0	0	1	0
PDM operation:	⇓				⇓	⇓
Dual chromosome:	0	1	0	0	0	1

Fig. 4. Example operation for the first probability-based PDM operator where $p_{\min} = 0.1$ and $p_{\max} = 0.9$.

Invalid PDM operations are unable to improve the performance of PDGA. In this section, a new probability-based PDM operator is proposed for improving the validity of the original PDM operator. In this PDM operator, the dual chromosome will be created by deciding for each bit of the primal chromosome string whether to translate to its complement according to a mapping probability of that bit.

Now, the question to be answered is how to set the mapping probability of each gene locus in a chromosome string. Here, we use some statistical information over the population to calculate the mapping probability of a gene locus. Let f_{ki} denote the frequency of k's in the alleles in gene locus $i (i = 1, 2, \ldots, L)$ over the population, where k is the allele value for the gene locus, and L is the chromosome length. In the binary-encoded space, $f_{1i} + f_{0i} = 1$, $0 \leq f_{0i}, f_{1i} \leq 1$, and f_{1i} can be regarded as the tendency to approach "1" for gene locus i over the population. If each f_{1i} tends to equal to 1 or 0, then the whole population is converging to one solution. Thus, a statistical vector $\{f_{11}, f_{12}, \ldots, f_{1L}\}$ can express the convergence degree of the population from the gene level. Let $p(i)$ denote the mapping probability of gene locus i in a chromosome, and, thus, a probability vector $\{p(1), p(2), \ldots, p(L)\}$ can be shared by the chromosomes, which are selected from the current population to execute the PDM operation. Based on the statistical vector $\{f_{11}, f_{12}, \ldots, f_{1L}\}$, two different methods can be considered to calculate the probability vector $\{p(1), p(2), \ldots, p(L)\}$ in this paper.

The first one is a mapping scheme to avoid the population convergence, which is calculated as

$$p(i) = p_{\min} + (p_{\max} - p_{\min}) \times |1 - 2 \cdot f_{1i}| \qquad (1)$$

where $i = 1, 2, \ldots, L$, $|y|$ denotes the absolute value of y, and $p_{\min}$ and $p_{\max}$ denote the minimum and maximum mapping probabilities for a gene locus, respectively. Obviously, we can see that $p(i)$ can achieve the minimum value $p_{\min}$ when f_{1i} is equal to 0.5, and achieve the maximum value $p_{\max}$ when f_{1i} is equal to 1 or 0. That is, the more the allele value in a gene locus converges, the larger its mapping probability is. Fig. 4 shows an example of applying this PDM operator to the same chromosome as in Fig. 3.

The second mapping scheme is to promote the population convergence, which is calculated as

$$p(i) = p_{\max} - (p_{\max} - p_{\min}) \times |1 - 2 \cdot f_{1i}| \qquad (2)$$

1's frequency in a locus:	0.1	0.4	0.8	0.3	0.7	0.2
PDM probability:	0.26	0.74	0.42	0.58	0.58	0.42
Generate a random number:	0.7	0.3	0.6	0.5	0.4	0.2
Whether mapping:	N	Y	N	Y	Y	Y
Primal chromosome:	1	1	0	0	1	0
PDM operation:		⇓		⇓	⇓	⇓
Dual chromosome:	1	0	0	1	0	1

Fig. 5. Example operation for the second probability-based PDM operator where $p_{\min} = 0.1$ and $p_{\max} = 0.9$.

where the parameter settings are the same as in (1). It can easily be seen that the more the allele value in a gene locus converges, the smaller its mapping probability is, because $p(i)$ can achieve the maximum value $p_{\max}$ when f_{1i} is equal to 0.5 and the minimum value $p_{\min}$ when f_{1i} is equal to 1 or 0. An example is also shown in Fig. 5, which is similar to the example in Fig. 4.

Based on the above description, each PDM operator can cause its respective influence upon the performance of the algorithm. The first probability-based PDM operator can make the primal chromosome have a chance of making a long jump to its complement in the search space, which can help improve the exploration capacity of PDGA. The second probability-based PDM operator can cause quick convergence of the population and hence can help PDGA exploit for a small area of the search space sufficiently. Obviously, the above two PDM operators are suitable for addressing different problems as a result of their different influence upon the algorithm's capacity.

C. Adaptive Probability-Based PDM Operator

Considering that the aforementioned two probability-based PDM schemes are always problem dependent, we propose an adaptive probability-based PDM operator. It employs both of the mapping schemes in the algorithm framework and uses a meta-Lamarckian learning strategy to decide which operator is chosen for executing one PDM operation with a probability.

Let $p_{\mathrm{sel},1}$ and $p_{\mathrm{sel},2}$ denote the probabilities of applying the first and second probability-based PDM operator to the chromosome that is selected for executing one PDM operation, respectively, where $p_{\mathrm{sel},1} + p_{\mathrm{sel},2} = 1$. Initially, both $p_{\mathrm{sel},1}$ and $p_{\mathrm{sel},2}$ are set to 0.5, which means giving a fair competition chance to each PDM operator. As each PDM operator always makes a biased mapping, the PDM operator that produces more improvements should be given greater selection probability. Inspired by the idea of Ong and Keane's work [18], a meta-Lamarckian learning strategy is integrated into PDGA and is elaborated as follows. Let η denote the improvement degree of a selected chromosome when one PDM operator is used on it, which can be calculated as

$$\eta = (f_{\mathrm{imp}} - f_{\mathrm{ini}})/f_{\mathrm{ini}} \tag{3}$$

where f_{imp} and f_{ini} are the fitness of the chromosome after and before executing one PDM operation, respectively. At each generation, the degree of improvement of each PDM operator is calculated when a predefined number of iterations are achieved, and then $p_{\mathrm{sel},1}$ and $p_{\mathrm{sel},2}$ are recalculated to proceed with the PDM operation in the next generation.

Assume that $\eta_1(t)$ and $\eta_2(t)$, respectively, denote the total improvement of the first and second probability-based PDM operators at generation t. The selection probabilities $p_{\mathrm{sel},1}(t+1)$ and $p_{\mathrm{sel},2}(t+1)$ at generation $(t+1)$ can be calculated by the following formulas:

$$p_{\mathrm{sel},1}(t+1) = \frac{p_{\mathrm{sel},1}(t) + \Delta \cdot \eta_1(t)}{p_{\mathrm{sel},1}(t) + \Delta \cdot \eta_1(t) + p_{\mathrm{sel},2}(t) + \Delta \cdot \eta_2(t)} \tag{4}$$

$$p_{\mathrm{sel},2}(t+1) = \frac{p_{\mathrm{sel},2}(t) + \Delta \cdot \eta_2(t)}{p_{\mathrm{sel},1}(t) + \Delta \cdot \eta_1(t) + p_{\mathrm{sel},2}(t) + \Delta \cdot \eta_2(t)} \tag{5}$$

where Δ signifies the relative degree of improvement influencing the selection probability. With the above discussion, the proposed adaptive probability-based PDM operator is summarized by the pseudocode in Fig. 6.

```
Procedure adaptive probability-based PDM operator
begin
  if p_sel,1 and p_sel,2 are not initialized then
    set p_sel,1 = p_sel,2 = 0.5;
  if no chromosomes in D(t) have executed PDM
    operations then
    calculate the probability vectors of the two
      probability-based PDM operators;
    set η1 = η2 = 0;
  if random() < p_sel,1 then
    x' :=createDualChromByFirstProbVector(x);
    evaluateChromosome(x');
    if a replacement condition is met then
      replace x in P(t) with x';
    update(η1);
  else
    x' :=createDualChromBySecondProbVector(x);
    evaluateChromosome(x');
    if a replacement condition is met then
      replace x in P(t) with x';
    update(η2);
  if all chromosomes in D(t) have executed PDM
    operations then
    recalculate(p_sel,1, p_sel,2);
end
Denotations:
  D(t): the set of chromosomes selected for executing
    PDM operations at generation t
  random(): a pseudo-random number between 0 and 1
  Other denotations are the same as those in Fig. 2
```

Fig. 6. Pseudocode for the adaptive probability-based PDM operator.

From the above discussion, the two different mapping operators may not only cooperate to promote each individual's effort but also compete with each other to achieve a greater selection probability in the running process of the adaptive probability-based PDM operator. To promote competition, their selection probabilities can be recalculated according to a meta-Lamarckian learning mechanism, where the PDM operator with a higher fitness improvement is rewarded with a higher chance of being chosen for the subsequent operations.

D. Adaptive Dominant Replacement Scheme

In addition to the above adaptive probability-based PDM scheme, instead of using a strict dominant replacement strategy

in the original PDGA algorithm, where only a dual chromosome that has a better fitness can replace the primal chromosome, an adaptive dominant replacement method is also proposed in this paper. In this adaptive scheme, a primal chromosome in the current population can be replaced with its dual with a probability given by the following formula:

$$p_{\text{acc}} = \begin{cases} 1, & \text{if } \Delta f \geq 0 \\ e^{k\Delta f/(f_{\text{best}}+\delta)}, & \text{else} \end{cases} \tag{6}$$

where $\Delta f = f(x') - f(x)$ is the difference between the fitness of the primal chromosome x and its dual x', f_{best} is the best fitness among the fitness values of the population, k is a normalization factor, and δ is a very small positive number. It can be observed from (6) that this adaptive dominance scheme can accept an inferior dual chromosome, which enables the algorithm to be highly explorative.

In the original PDGA, only the primal chromosomes need to be recorded in the population, and the dual representation of a primal chromosome is always unique. Thus, the original PDGA can be taken as working on a pseudo-diploid of primal–dual chromosomes considering that the encoding is based on a single-stranded chromosome instead of double stranded, as in a deoxyribonucleic acid (DNA) molecule. In contrast, in PDGA with probability-based PDM operators, there are many different representations for a chromosome's complement. This mechanism is similar to polyploidy in nature, but a pseudo-polyploid encoding scheme is actually used, because the dual of a primal chromosome need not be recorded physically in the population. Thus, it can be seen that the proposed probability-based PDM operators in this paper can be expected to help the algorithm explore the search space more efficiently and adapt to more diverse environmental dynamics than the original PDM operator.

IV. Algorithm Test Environments

To examine the performance of the proposed adaptive PDGA, a series of DOPs are constructed by the DOP generator proposed in [29] and [35] based on three stationary benchmark test problems, which are described below.

A. Stationary Test Problems

1) Knapsack Problem: The knapsack problem is a well-known NP-complete combinatorial optimization problem and has been well studied in the GA community. Here, a 100-item 0–1 knapsack problem is constructed with the weight and profit of each item randomly generated in the range of [1, 30] and the capacity of the knapsack set to half of the total weight of all items. The fitness of a feasible solution is the sum of the profits of the selected items. If a solution overfills the knapsack, then its fitness is set to the difference between the total weight of all items and the total weight of selected items, multiplied by a small factor 10^{-5} to make it in-competitive with those solutions that do not overfill the knapsack.

2) Royal Road Problem: The Royal Road problem is defined on a 100-bit binary string that consists of 25 contiguous building blocks of 4 bits. Each building block contributes four to the total fitness if its unitation (i.e., the number of ones inside the building block) is four; otherwise, it contributes zero. The overall fitness of an individual is calculated by the sum of contributions from all building blocks. The optimal fitness for this problem is 100.

3) Deceptive Problem: The deceptive functions are a family of functions in which there exists low-order building blocks that do not combine to form the higher-order building blocks. Here, a deceptive function that consists of 25 copies of the order-4 fully deceptive function DF2 is constructed for this paper. DF2 can be described as follows:

f(0000) = 28	f(0001) = 26	f(0010) = 24	f(0011) = 18
f(0100) = 22	f(0101) = 6	f(0110) = 14	f(0111) = 0
f(1000) = 20	f(1001) = 12	f(1010) = 10	f(1011) = 2
f(1100) = 8	f(1101) = 4	f(1110) = 6	f(1111) = 30.

This function has an optimum fitness of 750.

B. Constructing Dynamic Test Environments

The DOP generator proposed in [29] and [35] can generate dynamic environments from any binary-encoded stationary function $f(x)(x \in \{0,1\}^L)$ by a bit-wise exclusive-OR (XOR) operator. The environment is changed every τ generations. For each environmental period k, an XOR mask $\vec{M}(k)$ is incrementally generated as

$$\vec{M}(k) = \vec{M}(k-1) \oplus \vec{T}(k) \tag{7}$$

where $\oplus$ is the XOR operator (i.e., $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$), and $\vec{T}(k)$ is an intermediate binary template randomly created with $\rho \times L$ ones for the kth environmental period. For the first period $k = 1$, $\vec{M}(1) = \vec{0}$. Then, the population at generation t is evaluated as

$$f(\vec{x}, t) = f\left(\vec{x} \oplus \vec{M}(k)\right) \tag{8}$$

where $k = \lceil t/\tau \rceil$ is the environmental index.

The advantage of this XOR generator is that it can easily control the speed and severity of environmental changes. With this generator, the parameter τ controls the speed of changes, whereas $\rho \in (0.0, 1.0)$ controls the severity of changes. A bigger ρ means more severe changes, whereas a smaller τ means more frequent changes.

In this paper, the dynamic test environments are constructed using the above XOR DOP generator from the aforementioned three stationary functions. The dynamics parameter ρ is set to 0.1, 0.3, 0.5, 0.7, and 0.9, respectively, to examine the performance of algorithms in dynamic environments with different severities of changes: from slight changes ($\rho = 0.1$) to moderate variations ($\rho = 0.3$, 0.5 or 0.7) to intense changes ($\rho = 0.9$). The change speed parameter τ is set to 10, 100, and 200, respectively, to test each algorithm's capability of adapting

TABLE I
INDEX TABLE FOR THE ENVIRONMENTAL DYNAMIC SETTINGS

τ	Environmental Dynamics Index				
10	1	2	3	4	5
100	6	7	8	9	10
200	11	12	13	14	15
$\rho \rightarrow$	0.1	0.3	0.5	0.7	0.9

to dynamic environments under different searching stages.[1] In total, a series of 15 different dynamic problems are constructed from each stationary test problem. The dynamics parameter settings are summarized in Table I.

V. EXPERIMENTAL STUDY

A. Experimental Design

In this section, experiments are carried out to study the main characters of our proposed adaptive PDGAs and compare their performance with four existing peer algorithms: three complementary and dominance-based GAs and one recently developed GA with an elitism-based immigrant scheme. The following abbreviations represent these GAs considered in this paper.

1) DGA: The DGA proposed by Collard *et al.* in [5] (see Section II for more description).
2) oriPDGA: The original PDGA proposed in [29].
3) domGA: A diploid GA with adaptive domination change mechanism proposed by Uyar and Harmanci in [25]. In domGA, each individual has two chromosomes as a result of the diploid representation and a special scheme to determine the genotype-to-phenotype mapping. If the two alleles for a gene locus on two chromosomes are the same, then the phenotype is equal to that allele; otherwise, the phenotypic value is determined by the dominant factor of allele 1 over allele 0 for the corresponding locus on the chromosomes. For example, if the alleles on two genotype chromosomes are different for the gene at one location and the corresponding dominant factor is 0.8, then the phenotypic value for that location is 1 with a probability of 0.8 and 0 with a probability of 0.2. All dominant factors are set to 0.5 initially and are recalculated at the end of each generation based on the fitness values of the current population. In addition, a special two-stage crossover operator is also used in domGA, where one chromosome from each parent is first randomly selected to be copied into each of the two offspring, and then each offspring undergoes a uniform crossover that occurs between its own chromosomes.
4) EIGA: The GA with the elitism-based immigrants scheme studied in [31] and [32]. In EIGA, a set of immigrant chromosomes are generated by bit-wise mutating the elitist (the best chromosome) from the previous generation to replace the worst chromosomes in the population at each generation.
5) adaPDGA: The proposed PDGA with the probability-based PDM operator.

The following parameters are used in all the algorithms: The population size (pop_size) is set to 120 for DGA and domGA, but is set to 100 for EIGA, oriPDGA, and adaPDGA, because the immigrant ratio is set to 0.2 in EIGA per generation, and the number of chromosomes selected for executing the PDM operation per generation is set to 20 in PDGAs. The simple one-point crossover operation is used with the probability $pc = 0.6$ for DGA, oriPDGA, EIGA, and adaPDGA, whereas domGA uses its original two-stage crossover operation where the uniform crossover probability pc is also set to 0.6. For all the algorithms, the bit-wise mutation probability pm is set to 0.01. The specific parameters in adaPDGA are set as follows: $\Delta = 1$, $k = 1$, and $\delta = 0.001$. Other parameters in the peer algorithms are always the same as their original settings.

For each experiment of an algorithm on a test problem, 20 independent runs were executed with the same set of random seeds. For each run of an algorithm on a DOP, ten environmental changes were allowed, and the best-of-generation fitness was recorded per generation.

The overall performance of an algorithm is defined as the best-of-generation fitness averaged across the number of total runs and then averaged over the data gathering period, as formulated in the following:

$$\overline{F}_{BG} = \frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{N}\sum_{j=1}^{N} F_{BG_{ij}}\right) \tag{9}$$

where G is the number of generations (i.e., $G = 10 * \tau$), $N = 20$ is the total number of runs, and $F_{BG_{ij}}$ is the best-of-generation fitness of generation i of run j.

B. Experimental Study on the Effect of the PDM Operator

In our first set of experiments, we investigate the effect of the PDM operator on the performance of PDGAs on DOPs constructed in Section IV-B. To make a convenient description on the experiments, adaPDA1, adaPDA2, and adaPDA3 are used to denote adaPDGA with the first, second, and adaptive probability-based PDM operators, respectively. In addition, all the PDGAs use the dominant replacement scheme, which means that only the dual chromosome with a better fitness can replace the primal in the population. The experimental results with respect to the overall performance are presented in Table II and plotted in Fig. 7. The corresponding statistical results of comparing algorithms by the one-tailed t-test with 38 degrees of freedom at a 0.05 level of significance are given in Table III. In Table III, the t-test result regarding Alg. 1–Alg. 2 is shown as "$s+$," "$s-$," "$+$," or "$-$" when Alg. 1 is significantly better than, significantly worse than, insignificantly better than, or insignificantly worse than Alg. 2, respectively. From Tables II and III and Fig. 7, several results can be observed and are analyzed below.

[1]According to our preliminary experiments on stationary problems, all the algorithms are roughly at a quite early searching stage at generation 10, at a medium searching stage at generation 100, and at a late searching or converged stage at generation 200.

TABLE II
EXPERIMENTAL RESULTS OF PDGAS WITH DIFFERENT PDM OPERATORS ON THE DYNAMIC TEST PROBLEMS

Dynamics		Knapsack Problem				Royal Road Problem				Deceptive Problem			
τ	ρ	ori PDGA	ada PDGA1	ada PDGA2	ada PDGA3	ori PDGA	ada PDGA1	ada PDGA2	ada PDGA3	ori PDGA	ada PDGA1	ada PDGA2	ada PDGA3
10	0.1	1839.1	1842.4	1843.7	1842.7	40.8	41.8	47.7	46.0	573.1	571.4	582.4	582.7
10	0.3	1835.0	1835.7	1838.5	1838.1	27.7	28.8	32.2	31.6	512.5	521.3	526.5	528.6
10	0.5	1832.0	1832.7	1834.8	1834.6	25.8	27.5	28.0	28.5	502.0	510.9	516.0	513.4
10	0.7	1832.8	1823.5	1830.8	1829.1	27.5	27.5	27.1	27.4	511.0	513.7	516.1	516.2
10	0.9	1837.5	1808.0	1828.9	1824.2	40.5	31.6	29.7	30.1	550.0	541.9	546.0	544.2
100	0.1	1853.0	1858.0	1867.7	1868.0	87.1	86.6	92.5	92.0	699.4	700.9	698.7	704.7
100	0.3	1845.6	1849.4	1856.4	1856.8	68.7	67.2	78.3	77.3	653.9	657.1	673.9	668.5
100	0.5	1844.8	1846.0	1849.2	1850.4	61.9	61.3	66.7	68.4	636.2	646.7	661.7	655.5
100	0.7	1846.0	1850.1	1841.7	1852.6	69.0	69.0	62.3	76.7	652.9	657.8	673.5	665.5
100	0.9	1850.5	1861.4	1833.8	1861.5	87.3	87.2	64.8	91.2	699.7	700.1	705.8	703.5
200	0.1	1864.5	1865.1	1873.9	1874.9	93.4	93.5	96.3	96.3	720.9	719.3	709.4	720.0
200	0.3	1854.4	1856.1	1863.8	1864.1	81.2	79.8	88.7	88.2	691.3	694.5	692.9	699.8
200	0.5	1850.8	1852.9	1857.3	1858.1	77.5	75.9	82.5	83.5	681.3	687.2	688.5	696.4
200	0.7	1854.7	1855.6	1851.2	1860.7	81.5	81.2	80.6	88.4	692.3	694.7	696.8	705.7
200	0.9	1863.6	1868.1	1845.7	1871.8	93.8	93.4	83.3	95.9	719.0	719.2	714.6	722.4

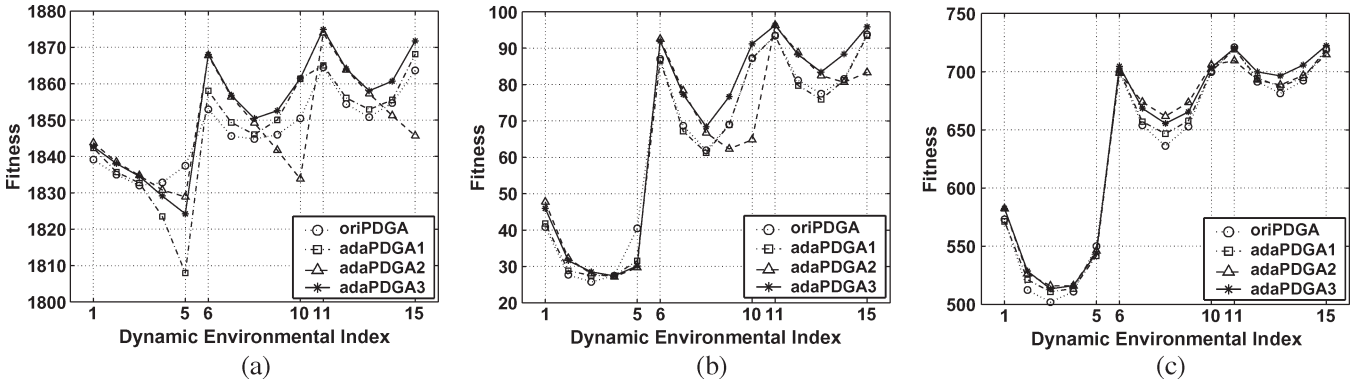

Fig. 7. Experimental results of PDGAs with different PDM operators on the dynamic test problems. (a) Knapsack. (b) Royal Road. (c) Deceptive.

TABLE III
STATISTICAL COMPARISON OF PDGAS WITH DIFFERENT PDM OPERATORS ON THE DYNAMIC TEST PROBLEMS

t-test Result	Knapsack Problem					Royal Road Problem					Deceptive Problem				
$\tau = 10$, $\rho \Rightarrow$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
adaPDGA3 − adaPDGA2	−	−	−	$s-$	$s-$	−	−	+	+	+	+	+	−	+	−
adaPDGA3 − adaPDGA1	+	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	−	$s-$	$s+$	$s+$	+	+	+
adaPDGA3 − oriPDGA	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	−	$s-$	$s+$	$s+$	$s+$	$s+$	−
adaPDGA2 − adaPDGA1	+	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	+	−	$s-$	$s+$	$s+$	$s+$	+	+
adaPDGA2 − oriPDGA	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	−	$s-$	$s+$	$s+$	$s+$	$s+$	−
adaPDGA1 − oriPDGA	$s+$	+	+	$s-$	$s-$	+	$s+$	$s+$	+	$s-$	−	$s+$	$s+$	+	$s-$
$\tau = 100$, $\rho \Rightarrow$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
adaPDGA3 − adaPDGA2	+	+	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$
adaPDGA3 − adaPDGA1	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
adaPDGA3 − oriPDGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
adaPDGA2 − adaPDGA1	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$
adaPDGA2 − oriPDGA	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	−	$s+$	$s+$	$s+$	$s+$
adaPDGA1 − oriPDGA	$s+$	$s+$	+	$s+$	$s+$	−	$s-$	−	−	−	$s+$	$s+$	$s+$	$s+$	+
$\tau = 200$, $\rho \Rightarrow$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
adaPDGA3 − adaPDGA2	+	+	+	$s+$	$s+$	−	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
adaPDGA3 − adaPDGA1	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	+	$s+$	$s+$	$s+$	$s+$
adaPDGA3 − oriPDGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	−	$s+$	$s+$	$s+$	$s+$
adaPDGA2 − adaPDGA1	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	−	$s-$	$s-$	$s-$	+	$s+$	$s-$
adaPDGA2 − oriPDGA	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s+$	$s+$	$s+$	$s-$
adaPDGA1 − oriPDGA	+	$s+$	$s+$	+	$s+$	+	$s-$	$s-$	−	$s-$	$s-$	$s+$	$s+$	$s+$	+

First, adaPDGA3 always outperforms the other algorithms on most of the dynamic problems when $\tau = 100$ and 200, as indicated in the relevant t-test results in Table III. The reason lies in that the two different PDM operators can help the algorithm adapt well to different environmental dynamics, which will be explained in detail in the latter experimental analysis, and the Lamarckian learning strategy help adaPDGA3 choose the suitable PDM operation to deal with the environmental changes. This result shows the efficiency of our proposed PDM operator in dynamic environments. When $\tau = 10$, adaPDGA3

underperforms some of the other algorithms on a few dynamic problems. This happens because when the environment changes very quickly, adaPDGA3 does not have enough time to adjust its PDM scheme efficiently.

Second, adaPDGA2 outperforms adaPDGA1 and oriPDGA when the change severity parameter ρ is small, but is beaten by them when the value of ρ is large on most dynamic Knapsack and Royal Road problems. This is because the second probability-based PDM operator in adaPDGA2 can make the population converge into a small area quickly and exploit this area sufficiently. When the environment changes slightly in terms of severity, the new optimum is very close to the previous one before a change. The sufficient exploitation can ensure that adaPDGA2 quickly achieves such changed optimum. In contrast, when the environment changes very significantly, the new optimum becomes far apart from the previous one. In these cases, it is obvious that adaPDGA2 cannot adapt well, since the converging population also makes adaPDGA2 lose enough exploration capacity. However, the situation seems a little different on some dynamic Deceptive problems where adaPDGA2 performs worse than adaPDGA1 and oriPDGA when ρ is small but performs better than them when ρ is large. The reason is that the deceptive attractor in the Deceptive problem can mislead the evolution process of adaPDGA2, and the XOR operator enables adaPDGA2 to escape from the deceptive attractor when the environment is subject to significant changes.

Third, adaPDGA1 performs better than adaPDGA2 on most dynamic Knapsack and Royal Road problems when $\rho = 0.7$ and $\rho = 0.9$. The reason lies in that the first probability-based PDM operator in adaPDGA1 enables the chromosomes to make a long jump in the search space. When a significant environmental change occurs, the XOR generator can strongly draw the chromosomes into the low-fitness area. Obviously, the PDM operation in adaPDGA1 is helpful to make the population quickly return to the high-fitness area. Moreover, it can be seen that adaPDGA1 outperforms oriPDGA on the dynamic knapsack problems, while underperforms the latter on most dynamic Royal Road problems when $\tau = 100$ or 200, although both of them adopt a similar PDM mechanism of avoiding the population convergence. This happens since the original PDM operator in oriPDGA could become noneffective once the population moves into a high-fitness area in the solution space in the Knapsack function, whereas the probability-based PDM operator in adaPDGA1 can cause destroying all the useful BBs, where each bit is equal to one, as achieved so far in the Royal Road function.

Fourth, oriPDGA beats adaPDGA1 and adaPDGA2 on some dynamic test problems only when $\rho = 0.9$, but underperforms adaPDGA3 on most dynamic test problems. This is because the original PDM function is designed as the maximum Hamming distance between a pair of primal–dual chromosomes, which makes oriPDGA adapt well to a very significant changing environment. In addition, the result of oriPDGA being beaten by adaPDGA3 with a high degree on most dynamic test problems confirms our expectation of the probability-based mechanism for PDGA in dynamic environments.

Finally, we also recorded the diversity of the population every generation to understand the effect of different PDM operators on the population diversity during the running of an algorithm. The diversity of the population at generation t in the kth run of an algorithm on a DOP is defined as

$$Div(k,t) = \frac{\sum_{i=0}^{pop\_size} \sum_{j \neq i}^{pop\_size} d(\vec{x}_i, \vec{x}_j)}{L \cdot pop\_size(pop\_size - 1)} \tag{10}$$

where L is the length of encoding, and $d(\vec{x}_i, \vec{x}_j)$ is the normalized Hamming distance between the ith and jth individuals in the population. The mean population diversity of an algorithm on a DOP at time t over 20 runs is calculated as

$$\overline{Div}(t) = \frac{1}{20} \sum_{k=1}^{20} Div(k,t). \tag{11}$$

The dynamic population diversity of algorithms against generations on DOPs with $\tau = 100$ and $\rho = 0.1$ or 0.9 is plotted in Fig. 8 for the ten environmental changes, where the data were averaged over 20 runs. From Fig. 8, it can be seen that oriPDGA always maintains the highest population diversity level on most dynamic problems since the PDM operation in oriPDGA always allows the chromosomes to make the longest jumps in the search space, whereas the appearance of the three adaPDGAs regarding the population diversity are different on different DOPs.

Compared with adaPDGA1, adaPDGA2 always maintains a lower population diversity level on all dynamic problems when $\rho = 0.1$. It is natural because the PDM operation in adaPDGA2 has an effect of encouraging convergence of the population. However, the situation becomes a little different when $\rho = 0.9$. The population diversity level of adaPDGA1 is much lower than that of adaPDGA2 on dynamic Knapsack and Royal Road problems. The reason lies in that the first probability-based PDM operator in adaPDGA1 can quickly map the population into a high-fitness area when the environmental change is very significant. Similar results can also be obtained from the fact that adaPDGA2 even has a higher population diversity than oriPDGA during the early search stage of each dynamic period when the value of ρ is 0.9. It is worthy to notice that adaPDGA1 maintains a higher population diversity than adaPDGA2 on dynamic Deceptive problems when $\rho = 0.9$. This happens because the population is always mapped into the area near the deceptive attractors in the deceptive function by the XOR generator when a significant environmental change occurs. It is impossible for adaPDGA2 to escape from the misleading of the deceptive attractor via its PDM operation, which causes its population always being in a very converged state.

Obviously, the above results show that PDM operators can affect the population diversity of PDGAs. Whether this effect is helpful or not depends on PDGAs and DOPs. For example, adaPDGA3 outperforms the other PDGAs on most dynamic problems although it just maintains a middle diversity level, whereas oriPDGA can maintain a higher diversity but a lower fitness at the same time on most dynamic problems when the value of ρ is small. In fact, similar results were also obtained in [31] and [32], where the random immigrant scheme cannot effectively improve the performance of GAs in some dynamic

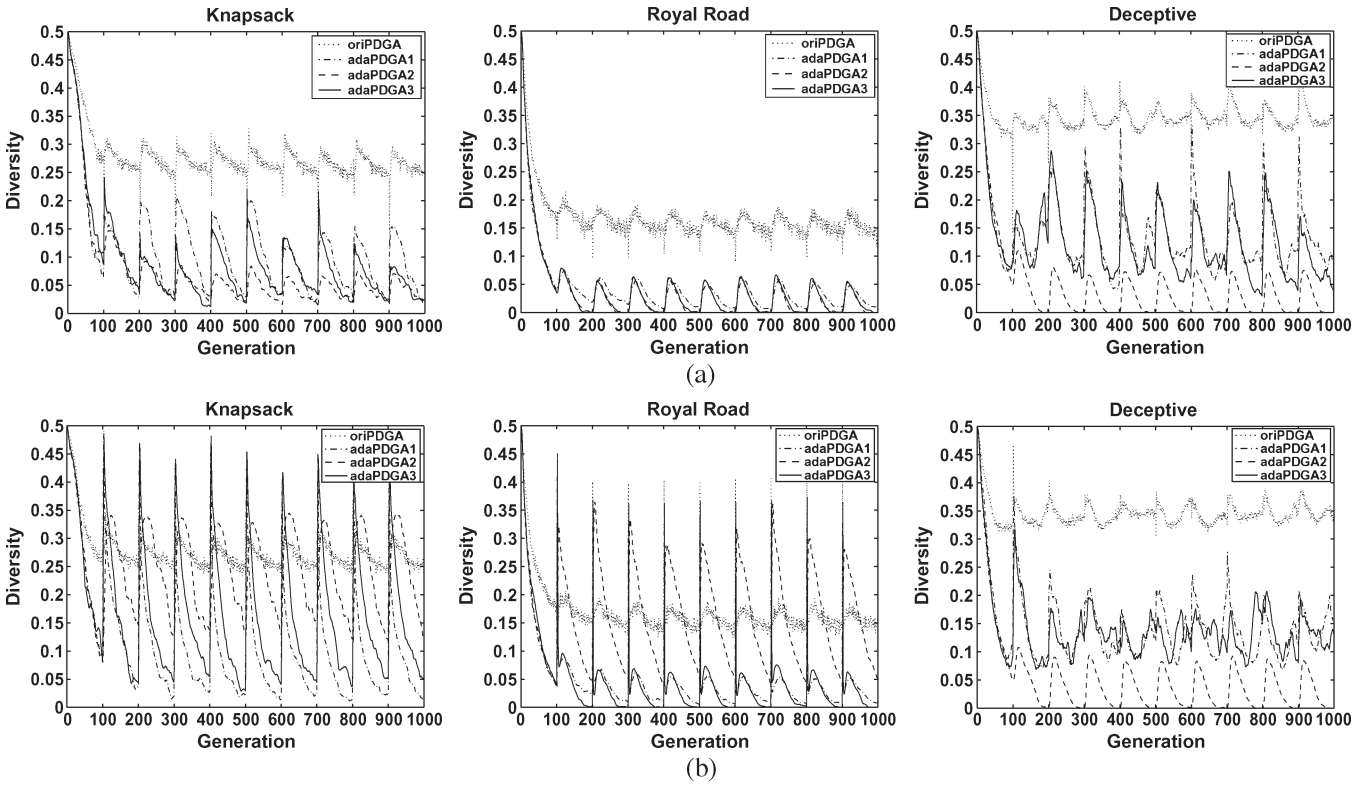

Fig. 8. Dynamic population diversity of PDGAs with different PDM operators on the dynamic test problems with $\tau = 100$ and ρ set to (a) $\rho = 0.1$ and (b) $\rho = 0.9$.

TABLE IV
EXPERIMENTAL RESULTS OF adaPDGAs WITH DIFFERENT REPLACEMENT SCHEMES ON THE DYNAMIC TEST PROBLEMS

Dynamics	Knapsack Problem			Royal Road Problem			Deceptive Problem		
ρ	adaPDGA_1	adaPDGA_2	t-test	adaPDGA_1	adaPDGA_2	t-test	adaPDGA_1	adaPDGA_2	t-test
0.1	1867.8	1868.8	$+$	92.3	92.5	$+$	703.3	701.9	$-$
0.3	1856.9	1857.9	$s+$	77.1	77.6	$+$	667.9	666.1	$-$
0.5	1850.6	1852.3	$s+$	68.1	69.7	$s+$	657.4	654.4	$s-$
0.7	1851.0	1852.4	$s+$	76.3	76.7	$+$	665.4	664.2	$-$
0.9	1862.7	1863.0	$+$	91.2	91.5	$+$	704.0	702.2	$-$

environments, although it can always help maintain a higher population diversity level.

C. Experimental Study on the Effect of the Replacement Scheme

In the above experiments, all the PDGAs use a dominant replacement scheme in the PDM operations. That is, the primal chromosome can be replaced by its dual only when the latter has a better fitness. To investigate the effect of different replacement methods on the PDM operator, we further carry out experiments on adaPDGA on the DOPs with $\tau = 100$ and ρ set to 0.1, 0.3, 0.5, 0.7, and 0.9, respectively. For the convenience of description, adaPDGA_1 and adaPDGA_2 denote adaPDGA with the dominant replacement scheme and the adaptive dominant replacement scheme, respectively, in the following experiments. The experimental results are presented in Table IV, where the t-test column denotes the one-tailed t-test results regarding adaPDGA_2 - adaPDGA_1 with 38 degrees of freedom at a 0.05 level of significance.

From Table IV, the following results can be observed. First, the replacement scheme does affect the performance of adaPDGA on DOPs, and the effect is problem dependent. In general, adaPDGA_2 always outperforms adaPDGA_1 on Knapsack and Royal Road problems, whereas adaPDGA_1 always outperforms adaPDGA2 on Deceptive problems. The reason lies in the intrinsic characteristics of these problems. AdaPDGA_2 can always maintain a higher population diversity level and has a stronger exploration capacity than adaPDGA_1. This is because AdaPDGA_2 can accept inferior dual chromosomes in the PDM operations, which helps adaPDGA_2 adapt better to the changing environment. This is the reason why adaPDGA_2 performs better than adaPDGA_1 on Knapsack and Royal Road problems. However, the Deceptive problem can mislead the evolution of algorithms due to the existence of deceptive attractors. The adaptive dominant replacement scheme can cause a lot of useless redundant searches, which greatly degrade the performance of adaPDGA_2. Although the dominant scheme can cause the population to be attracted by the deceptive attractor, the PDM operator can help adaPDGA_1 deal with this misleading to a certain extent.

Second, the degree of the effect of the replacement scheme on the performance of adaPDGA on DOPs depends on the value of ρ. For example, adaPDGA_2 significantly outperforms

TABLE V
EXPERIMENTAL RESULTS OF adaPDGA AND PEER GAs ON THE DYNAMIC TEST PROBLEMS

Dynamics		Knapsack Problem				Royal Road Problem				Deceptive Problem			
τ	ρ	DGA	domGA	EIGA	adaPDGA	DGA	domGA	EIGA	adaPDGA	DGA	domGA	EIGA	adaPDGA
10	0.1	1840.2	1810.3	1845.1	1842.6	38.2	34.3	55.3	46.2	558.6	597.2	635.5	574.8
10	0.3	1829.6	1805.8	1829.6	1836.9	27.1	25.9	37.9	32.3	505.4	534.5	573.5	524.1
10	0.5	1813.3	1796.8	1812.1	1835.0	23.6	24.0	31.0	28.1	491.4	515.3	554.8	513.6
10	0.7	1801.5	1782.3	1781.6	1830.0	23.5	23.4	29.7	27.1	494.7	515.1	559.9	513.6
10	0.9	1797.7	1770.8	1747.1	1827.5	29.0	26.2	35.2	30.4	542.5	542.4	607.4	537.5
100	0.1	1859.2	1860.3	1871.6	1869.3	86.7	83.8	90.1	92.3	698.4	677.7	702.4	703.0
100	0.3	1846.6	1847.8	1859.2	1857.2	66.6	60.2	77.7	77.9	652.5	635.1	691.4	667.4
100	0.5	1837.9	1843.2	1852.0	1852.3	53.5	48.2	66.9	69.4	635.2	609.9	688.1	656.4
100	0.7	1831.9	1837.3	1843.8	1851.9	61.4	43.7	59.9	76.6	648.9	616.7	697.1	664.2
100	0.9	1840.7	1828.8	1835.1	1862.9	81.9	44.4	53.1	91.1	696.6	663.8	701.9	713.8
200	0.1	1868.2	1868.0	1876.3	1875.9	93.2	91.3	94.7	96.2	714.2	690.4	703.8	719.7
200	0.3	1853.9	1854.8	1867.0	1864.6	80.4	74.6	86.7	88.7	687.3	668.1	697.7	699.6
200	0.5	1846.3	1849.1	1861.3	1858.8	68.3	61.6	79.3	84.5	678.6	653.0	693.8	700.3
200	0.7	1843.4	1843.8	1860.9	1861.8	78.1	53.4	73.7	88.3	690.5	658.6	702.9	706.7
200	0.9	1858.7	1837.6	1855.9	1872.0	91.2	49.0	67.7	95.7	716.1	694.0	718.4	721.8

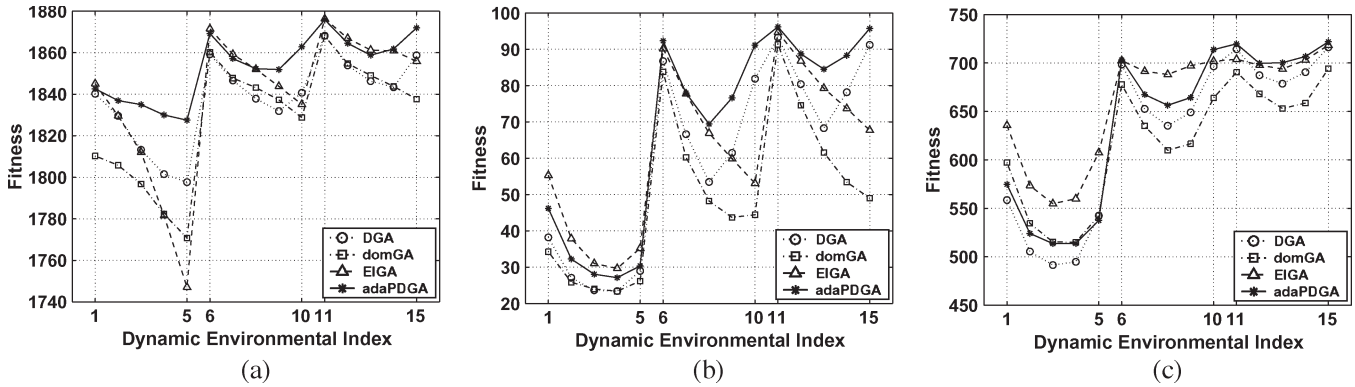

Fig. 9. Experimental results of adaPDGA and peer GAs on the dynamic test problems. (a) Knapsack. (b) Royal Road. (c) Deceptive.

adaPDGA_1 when $\rho = 0.3$, 0.5, and 0.7 while performing only a little better than adaPDGA_1 when $\rho = 0.1$ and 0.9 on the Knapsack problems. The reason is that the effects of the two replacement schemes upon the performance of the algorithm are very small when the environment changes very significantly or very slightly since the PDM operator in adaPDGAs can quickly map the chromosomes into the area near the new optimum in these cases. Similar results can be obtained on the Royal Road and Deceptive problems.

D. Experimental Study on Comparing adaPDGA With Peer GAs on DOPs

In the final set of experiments, we compare the performance of adaPDGA that combines the adaptive probability-based PDM operator and the adaptive dominant replacement scheme with several other peer GAs proposed in the literature on the DOPs constructed in Section IV-B. The peer GAs are EIGA, DGA, and domGA. The experimental results are presented in Table V and plotted in Fig. 9. The corresponding statistical results are given in Table VI. From Fig. 9 and Tables V and VI, several results can be observed and are analyzed as follows.

First, adaPDGA always significantly outperforms other peer GAs on most dynamic test problems, particularly when $\tau = 200$. This is because the first probability-based PDM scheme can enhance the exploration capability of adaPDGA and make it adapt well to significant environmental changes, whereas the second probability-based PDM scheme can improve adaPDGA's performance in slightly changing environments since it can enable adaPDGA to execute sufficient exploitation in the higher fitness area of the search space. The good performance of adaPDGA over other peer GAs confirms our expectation that the adaptive learning strategy can balance the exploration and exploitation capacity well for adaPDGA in dynamic environments. However, adaPDGA underperforms EIGA on dynamic Royal Road and Deceptive problems when $\tau = 10$, which shows that the adaptive probability-based PDM operator requires more "energy" to choose more effective PDM operations.

Similar results can be observed in the dynamic behavior of GAs, as plotted in Figs. 10–12, where τ is set to 100, and ρ is set to 0.1, 0.5, and 0.9, respectively. From these figures, it is easily observed that adaPDGA can always maintain a higher fitness level than domGA and DGA can do for all environmental periods on all dynamic problems. The experimental results that adaPDGA is beaten by EIGA with respect to the best-of-generation fitness will be explained in the experimental analysis later on. When $\rho = 0.1$ and $\rho = 0.9$, adaPDGA performs better for the dynamic periods than it does for the stationary period (the first 100 generations), whereas the dynamic behavior of adaPDGA for each dynamic period is almost the same as that for the stationary period when $\rho = 0.5$.

TABLE VI
STATISTICAL COMPARISON OF adaPDGA AND PEER GAs ON THE DYNAMIC TEST PROBLEMS

t-test Result	Knapsack Problem					Royal Road Problem					Deceptive Problem				
$\tau = 10$, $\rho \Rightarrow$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
adaPDGA − EIGA	−	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
adaPDGA − domGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s-$	−	−	$s-$
adaPDGA − DGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	−
EIGA − domGA	$s+$	$s+$	$s+$	−	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIGA − DGA	+	−	−	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
domGA − DGA	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	+	−	$s-$	$s+$	$s+$	$s+$	$s+$	−
$\tau = 100$, $\rho \Rightarrow$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
adaPDGA − EIGA	$s-$	$s-$	+	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	+	$s-$	$s-$	$s-$	$s+$
adaPDGA − domGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
adaPDGA − DGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIGA − domGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIGA − DGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$
domGA − DGA	+	+	$s+$	$s+$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$
$\tau = 200$, $\rho \Rightarrow$	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
adaPDGA − EIGA	−	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
adaPDGA − domGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
adaPDGA − DGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIGA − domGA	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
EIGA − DGA	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s-$	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$
domGA − DGA	−	+	$s+$	+	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$	$s-$

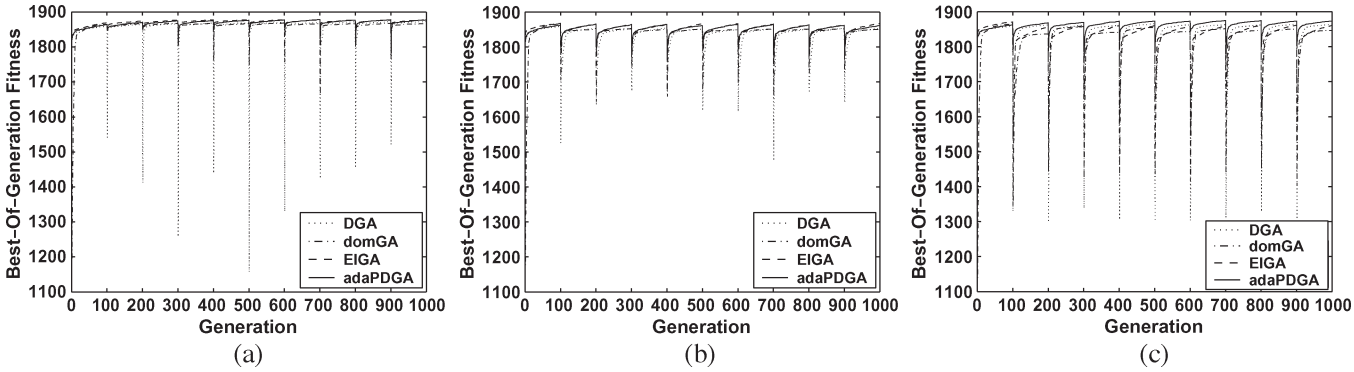

Fig. 10. Dynamic performance of adaPDGA and peer GAs on dynamic Knapsack problems with $\tau = 100$ and ρ set to (a) $\rho = 0.1$, (b) $\rho = 0.5$, and (c) $\rho = 0.9$.

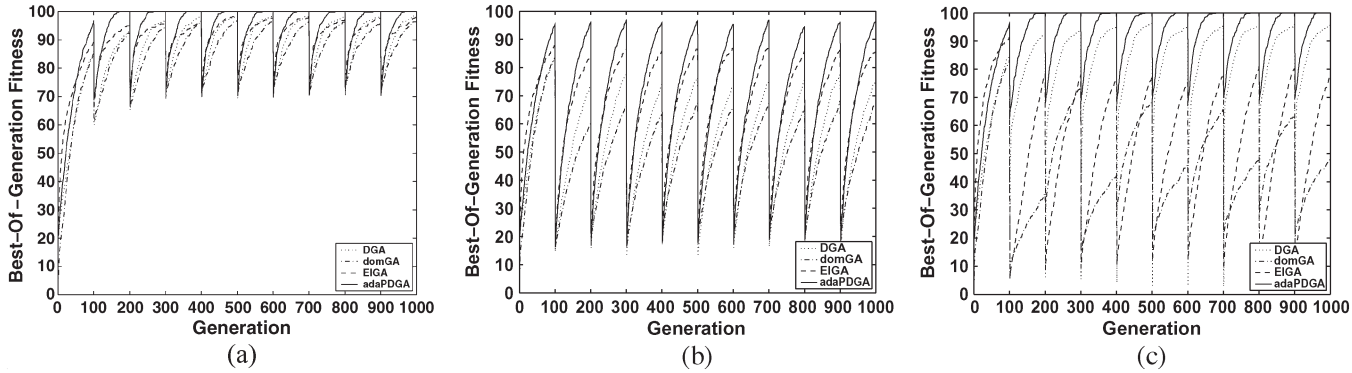

Fig. 11. Dynamic performance of adaPDGA and peer GAs on dynamic Royal Road problems with $\tau = 100$ and ρ set to (a) $\rho = 0.1$, (b) $\rho = 0.5$, and (c) $\rho = 0.9$.

Second, on dynamic Knapsack and Royal Road problems, EIGA performs better when the change severity ρ is not very large. This happens because the elitism-based immigrants scheme in EIGA can introduce higher fitness chromosomes into the population on dynamic Knapsack and Royal Road problems when the environment changes slightly. However, on dynamic Deceptive problems, the situation becomes a little different. On dynamic Deceptive problems, EIGA performs better than the other algorithms, including adaPDGA, when the value of ρ is very large, but performs worse than adaPDGA when $\rho = 0.1$. The reason lies in the fact that the deceptive attractor can mislead EIGA's evolution, because the immigrants are only generated close to the elitist chromosome in the current population. When the environment is subject to significant changes $(\rho = 0.9)$, the XOR operation may enable EIGA to jump out from the deceptive attractor.

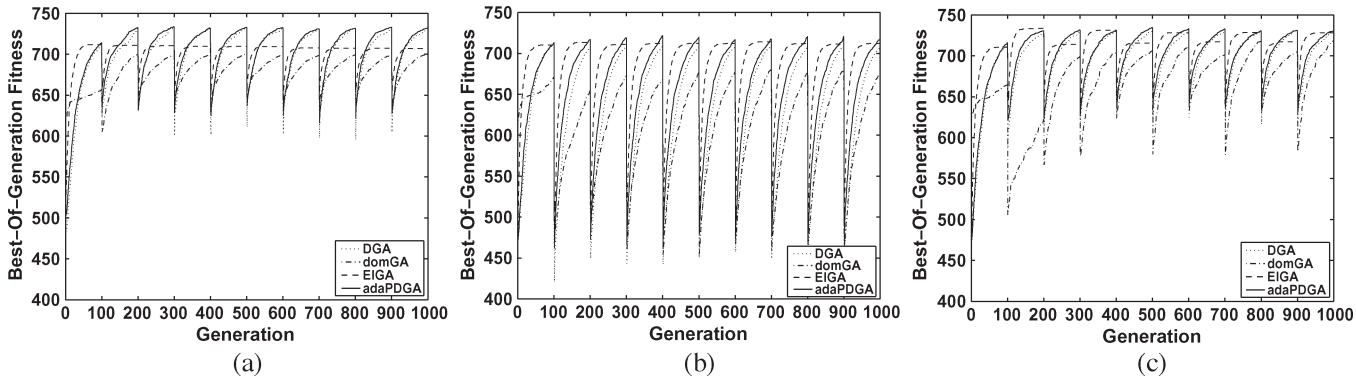

Fig. 12. Dynamic performance of adaPDGA and peer GAs on dynamic Deceptive problems with $\tau = 100$ and ρ set to (a) $\rho = 0.1$, (b) $\rho = 0.5$, and (c) $\rho = 0.9$.

The dynamic behavior of GAs in Figs. 10–12 can further indicate the aforementioned experimental results. For example, on dynamic Deceptive problems, EIGA performs better than other GAs for the stationary period, but is greatly beaten by adaPDGA and DGA for the dynamic periods after several generations when $\rho = 0.1$. When $\rho = 0.9$, EIGA's performance landscape is sort of switching between odd and even environmental periods. This is because after the stationary period, for the following odd period, the environment is in fact greatly returned or repeated from the previous odd period given $\rho = 0.9$.

Third, the performance of domGA is not exciting on most dynamic test problems, which can further be observed in the dynamic behavior of domGA in Figs. 10–12. The domGA performs worse for the dynamic periods than it does for the stationary period except on dynamic Knapsack and Royal Road problems when $\rho = 0.1$ and on dynamic Deceptive problems when $\rho = 0.1$ and $\rho = 0.9$. This happens because although the diploidy individual in domGA can store much redundant information, the implicit use of information cannot directly improve the performance of domGA in dynamic environments.

Fourth, DGA performs better than domGA and EIGA on dynamic test problems only when $\rho = 0.9$. The reason lies in the fact that a chromosome can jump to its dual that has a maximum distance away to it via mutating the meta-bit in DGA. However, the blindness in mutating the meta-bit results in that DGA is significantly beaten by adaPDGA. In addition, DGA's performance in slightly changing environments indicates that the effect of the dualism mechanism in DGA is limited. This can further be confirmed by the dynamic behavior of DGA in Figs. 10–12. DGA's performance landscapes are almost the same as adaPDGA's for each dynamic period on dynamic Knapsack and Deceptive problems when $\rho = 0.9$, whereas adaPDGA significantly beats DGA on most dynamic problems when ρ is not very large.

Finally, the environmental parameters affect the performance of algorithms. The performance of all algorithms increases when the value of τ increases from 10 to 100 to 200. It is easy to understand since algorithms have more time to find better solutions before the next change with the increment of the value of τ. The effect of the changing severity parameter ρ is different. For example, when τ is fixed, adaPDGA's performance curve is always lower when the environment changes with a middle severity degree than that when $\rho = 0.1$ and $\rho = 0.9$.

VI. Conclusion and Future Work

In this paper, a variant of GA, i.e., PDGA, has been discussed, and its key operator, i.e., the PDM operator, has further been investigated to improve its robustness and adaptability in dynamic environments. In the improved scheme, two different probability-based PDM methods, where the probability of each allele in the chromosome string that takes part in the PDM operation is calculated using the statistical information of the distribution of alleles in the corresponding gene locus over the population, are probabilistically selected to execute one PDM operation. A learning mechanism is introduced to give the more effective PDM operator a greater chance to be selected for later operations. In addition, an adaptive dominant replacement scheme is also introduced into the proposed PDGAs to enhance their exploration capability to adapt well to environmental changes.

From the experimental results based on a series of dynamic test problems constructed from three stationary benchmark problems, the following conclusions can be drawn on the dynamic test problems.

First, the probability-based mechanism can significantly improve the performance of the PDM operator in PDGAs in dynamic environments. On most tested DOPs, adaPDGA outperforms oriPDGA.

Second, the two probability-based PDM operators have different effects in different dynamic environments. The first probability-based PDM operator often performs better when the environment is subject to significant changes, whereas the second operator performs better when the environment changes slightly. The adaptive learning strategy can help adaPDGA execute a robust PDM operation since it employs both of the PDM operators under the mechanism of cooperation and competition.

Third, the effect of different replacement schemes is problem dependent. The adaptive dominant replacement scheme takes effect on the dynamic Knapsack and Royal Road problems, but the dominant replacement scheme is suitable for the dynamic Deceptive problems.

Fourth, the environmental dynamics can affect the performance of algorithms. In our experiments, algorithms perform better with the increase of the frequency of changes, and the effect of the severity of changes is problem dependent, which has also been observed before by some studies in the literature.

In summary, our experimental results show that the proposed adaPDGA with the adaptive PDM operator and the adaptive dominant replacement scheme seems a good optimizer for DOPs.

There are several future works relevant to this paper. It is a straightforward work to extend the idea of the PDM mechanism to GAs with other encodings and examine their performance in dynamic environments. For example, some promising results have been shown in the preliminary experiments of extending the idea to GAs for dynamic problems with order encoding in [15]. We believe that the well-designed PDM scheme should also improve the performance of GAs for continuous DOPs. To hybridize the proposed adaptive PDGAs with other approaches developed into GAs for DOPs, such as the memory and multipopulation schemes, will be another interesting future work. Moreover, the number of chromosomes selected for PDM operations per generation can vary according to the change of valid PDM operations in the original algorithm, but was set to a fixed constant in this paper. Then, it is valuable to further investigate how much the number of PDM operations per generation affects the performance of adaPDGA. Finally, we have only chosen a reasonable set of relevant parameters in the proposed algorithms and have not made any effort in finding the best setting for the proposed algorithms, which is left as a future research.

ACKNOWLEDGMENT

The author would like to thank the anonymous associate editor and reviewers for their thoughtful suggestions and constructive comments.

REFERENCES

[1] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1875–1882.
[2] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Norwell, MA: Kluwer, 2002.
[3] J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Proc. 4th Int. Conf. Adaptive Comput. Des. Manuf.*, 2000, pp. 299–308.
[4] H. G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environment," Naval Res. Lab., Washington, DC, Tech. Rep. AIC-90-001, 1990.
[5] P. Collard, C. Escazut, and A. Gaspar, "An evolutionary approach for time dependant optimization," *Int. J. Artif. Intell. Tools*, vol. 6, no. 4, pp. 665–695, 1997.
[6] Q. Chen and S. Guan, "Incremental multiple objective genetic algorithms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 3, pp. 1325–1334, Jun. 2004.
[7] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 425–442, Oct. 2004.
[8] E. Gelenbe, P. Liu, and J. Lainé, "Genetic algorithms for route discovery," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1247–1254, Dec. 2006.
[9] D. E. Goldberg and R. E. Smith, "Nonstationary function optimization using genetic algorithms with dominance and diploidy," in *Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp. 59–68.
[10] C. K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.
[11] J. J. Grefenstette, "Genetic algorithms for changing environments," in *Proc. 2nd Int. Conf. Parallel Problem Solving From Nature*, 1992, pp. 137–144.
[12] B. S. Hadad and C. F. Eick, "Supporting polyploidy in genetic algorithms using dominance vectors," in *Proc. 6th Int. Conf.* Evol. Program., 1997, pp. 223–234.
[13] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments—A survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.
[14] J. Lewis, E. Hart, and G. Ritchie, "A comparison of dominance mechanisms and simple mutation on non-stationary problems," in *Proc. 4th Int. Conf. Parallel Problem Solving From Nature*, 1998, pp. 139–148.
[15] L. Liu, D. Wang, and H. Wang, "A new dual scheme for genetic algorithm in dynamic environments," in *Proc. Chin. Control Decision Conf.*, 2008, pp. 71–74.
[16] R. W. Morrison and K. A. De Jong, "Triggered hypermutation revisited," in *Proc. IEEE Congr. Evol. Comput.*, 2000, pp. 1025–1032.
[17] K. P. Ng and K. C. Wong, "A new diploid scheme and dominance change mechanism for non-stationary function optimization," in *Proc. 6th Int. Conf. Genetic Algorithms*, 1995, pp. 159–166.
[18] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.
[19] F. Oppacher and M. Wineberg, "The shifting balance genetic algorithm: Improving the GA in a dynamic environment," in *Proc. Genetic Evol. Comput. Conf.*, 1999, vol. 1, pp. 504–510.
[20] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 440–458, Aug. 2006.
[21] C. Ryan, "Diploidy without dominance," in *Proc. 3rd Nordic Workshop Genetic Algorithms*, 1997, pp. 63–70.
[22] R. E. Smith, "Diploid genetic algorithms for search in time varying environments," in *Proc. Annu. Southeast Regional Conf. ACM*, 1987, pp. 175–179.
[23] W. Sheng, S. Swift, L. Zhang, and X. Liu, "A weighted sum validity function for clustering with a hybrid niching genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 6, pp. 1156–1167, Dec. 2005.
[24] Y. Takahashi, "A mathematical framework for solving dynamic optimization problems with adaptive networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 404–416, Aug. 1998.
[25] A. S. Uyar and A. E. Harmanci, "A new population based adaptive dominance change mechanism for diploid genetic algorithms in dynamic environments," *Soft Comput.*, vol. 9, no. 11, pp. 803–815, Nov. 2005.
[26] F. Vavak, C. Fogarty, and K. Jukes, "Adaptive combustion balancing in multiple burner boilers using a genetic algorithm with variable range of local search," in *Proc. 7th Int. Conf. Genetic Algorithms*, 1996, pp. 719–726.
[27] H. Wang and D. Wang, "An improved primal–dual genetic algorithm for optimization in dynamic environments," in *Proc. 13th Int. Conf. Neural Inf. Process.*, 2006, pp. 836–844. Part III.
[28] H. Wang, D. Wang, and S. Yang, "Triggered memory-based swarm optimization in dynamic environments," in *EvoWorkshops: Appl. Evol. Comput.*, 2007, LNCS, vol. 4448, pp. 637–646.
[29] S. Yang, "Non-stationary problem optimization using the primal–dual genetic algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2003, vol. 3, pp. 2246–2253.
[30] S. Yang, "Associative memory scheme for genetic algorithms in dynamic environments," in *Applications of Evolutionary Computing*, vol. 3907. Berlin, Germany: Springer-Verlag, 2006, pp. 788–799.
[31] S. Yang, "Genetic algorithms with elitism-based immigrants for changing optimization problems," in *EvoWorkshops: Appl. Evol. Comput.*, 2007, LNCS, vol. 4448, pp. 627–636.
[32] S. Yang, "Genetic algorithms with memory- and elitism-based immigrants in dynamic environments," *Evol. Comput.*, vol. 16, no. 3, pp. 385–416, 2008.
[33] S. Yang and H. Richter, "Hyper-learning for population-based incremental learning in dynamic environments," in *Proc. IEEE Congr. Evol. Comput.,* 2009, to be published.
[34] S. Yang and R. Tinos, "Hyper-selection in dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3185–3192.
[35] S. Yang and X. Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems," *Soft Comput.*, vol. 9, no. 11, pp. 815–834, Nov. 2005.
[36] S. Yang and X. Yao, "Population-based incremental learning with associative memory for dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 542–561, Oct. 2008.
[37] D. Z. Zhang, A. Anosike, and M. K. Lim, "Dynamically integrated manufacturing systems (DIMS)—A multiagent approach," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 5, pp. 824–850, Sep. 2007.

Hongfeng Wang received the B.Sc. degree in industrial engineering and the M.Sc. and Ph.D. degrees in systems engineering from Northeastern University, Shenyang, China, in 2001, 2004, and 2007, respectively.

He is currently a Lecturer with the Institute of Systems Engineering, Northeastern University, Shenyang, China. His major research interests include evolutionary algorithms, memetic computing, computational intelligence in dynamic environments, production scheduling, and supply chain management.

Shengxiang Yang (M'00) received the B.Sc. and M.Sc. degrees in automatic control and the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China, in 1993, 1996, and 1999, respectively.

From October 1999 to October 2000, he was a Postdoctoral Research Associate with the Algorithm Design Group, Department of Computer Science, King's College London, London, U.K. He is currently a Lecturer with the Department of Computer Science, University of Leicester, Leicester, U.K. He has over 90 publications. He has given invited keynote speeches in several international conferences and coorganized several workshops and special sessions in conferences. He serves as the Area Editor, Associate Editor, or Editorial Board Member for four international journals. He has coedited several books and conference proceedings and co-guest-edited several journal special issues. His major research interests include evolutionary algorithms, swarm intelligence, meta-heuristics and hyper-heuristics, artificial neural networks, computational intelligence in dynamic and uncertain environments, scheduling, network flow problems and algorithms, and real-world applications.

Dr. Yang is a Member of the Association for Computing Machinery (ACM) Special Interest Group on Genetic and Evolutionary Computation (SIGEVO). He is a Member of the Task Force on Evolutionary Computation in Dynamic and Uncertain Environments, Evolutionary Computation Technical Committee, IEEE Computational Intelligence Society.

W. H. Ip received the M.Sc. degree from Cranfield University, Bedfordshire, U.K., in 1983, the M.B.A. degree from Brunel University, Middlesex, U.K., in 1989, and the Ph.D. degree in manufacturing engineering from Loughborough University, Loughborough, U.K., in 1993.

He is currently an Associate Professor with the Department of Industrial and Systems Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong. Over the span of 20 years of work in industry, education, and consulting, he has published more than 100 international journals and conference articles in the area of computer and operational research, artificial intelligence, and decision-support systems. His research interests include the modeling of the application of intelligent algorithms in planning and scheduling, evolutionary computing, and computer visions.

Dr. Ip is a Member of the Hong Kong Institution of Engineers, the Institution of Engineering and Technology, and the Institution of Mechanical Engineers.

Dingwei Wang received the M.Sc. degree in systems engineering from Huazhong University of Science and Technology, Wuhan, China, in 1984 and the Ph.D. degree in control theory and application from Northeastern University, Shenyang, China, in 1993.

From 1994 to 1995, he was a Postdoctoral Fellow with the North Carolina State University, Raleigh. He is currently a Professor with the Institute of Systems Engineering, Northeastern University, Shenyang, China. He has published more than 300 papers in international or domestic journals including IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, IEEE TRANSACTIONS ON NEURAL NETWORKS, *Sciences in China*, etc. His research interests include the modeling of the optimization of complex systems, evolutionary computation, and soft computing.

Dr. Wang is a Member of the Chinese Association Automation and the Chinese Association of Operations Research. He serves on the editorial board of *Fuzzy Optimization and Decision Making* and several Chinese journals.