Learning the Dominance in Diploid Genetic Algorithms for Changing Optimization Problems

Shengxiang Yang

***Abstract*— Using diploid representation with dominance scheme is one of the approaches developed for genetic algorithms to address dynamic optimization problems. This paper proposes an adaptive dominance mechanism for diploid genetic algorithms in dynamic environments. In this scheme, the genotype to phenotype mapping in each gene locus is controlled by a dominance probability, which is learnt adaptively during the searching progress. The proposed dominance scheme is experimentally compared to two other schemes for diploid genetic algorithms. Experimental results validate the efficiency of the dominance learning scheme.**

I. Introduction

Genetic algorithms (GAs) are often used to solve stationary optimization problems. However, in real world many optimization problems are actually dynamic, where changes may occur over time. For dynamic optimization problems (DOPs), the aim of GAs is no longer to locate an optimal solution quickly and precisely but to track the moving optimum with time. This seriously challenges traditional GAs since once GAs are converged to a solution, it is hard for them to adapt to the new environment when a change occurs. In order to address DOPs, researchers have developed several approaches into GAs to enhance their performance [3], of which using memory schemes is one major approach.

The basic principle of memory schemes is to store and reuse useful information. The information may be stored implicitly [6], [7], [8] or explicitly [2]. For implicit memory schemes, GAs use redundant representations in genotype to store good (partial) solutions to be reused later. A typical example is the *diploid genetic algorithms (DGA)*, which was inspired by the diploid representations and dominance scheme in biology. DGAs have proved to be advantageous to address DOPs [6], [8], [9], [10]. For DGAs, a key factor is to develop good dominance schemes, which map the diploid genotype to phenotype. It is also claimed that diploidy with dominance changes is essential [7].

In this paper, an adaptive dominance scheme is proposed for DGAs for DOPs. In this scheme, the genotype to phenotype mapping is controlled by a *dominance probability vector*. The dominance probability vector is learnt toward the best individual of the current population during the searching progress and hence is adapted to the dynamic environment. Using the DOP generator proposed in [11], [12], a series of DOPs are constructed as the test environments and experiments are carried out to compare the proposed dominance scheme against two state-of-the-art dominance schemes: the Ng-Wong [8] and the additive [9] dominance schemes with dominance change [7] for DGAs. The experimental results validates the efficiency of the proposed dominance scheme for DGAs for DOPs.

The rest of this paper is outlined as follows. The next section briefly describes the framework of DGAs and several dominance schemes developed for DGAs for DOPs. Section III presents the proposed adaptive dominance scheme for DGAs in dynamic environments. The experimental study and analysis are presented in Section IV. Section V concludes this paper with discussions on relevant future work.

Shengxiang Yang is with the Department of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, United Kingdom, (email: s.yang@mcs.le.ac.uk).

II. Diploidy and Dominance

A. Framework of Diploid Genetic Algorithms

Most advanced organisms have a diploid or even multiploid chromosome structure. In the diploid structure, two set of *homologue chromosomes* are twisted together into a duplex deoxyribonucleic (DNA) structure. *Genes* are the smallest hereditary unit that contains genetic information and two or more alternative forms/values that genes occupying the same locus on homologue chromosomes can take are called *alleles*. The *genotype* of an individual consists of all the genes located on all the homologue chromosomes and those genes that are expressed form the *phenotype* of the individual. A *dominant* allele is always expressed while a *recessive* allele may be expressed only when both genes occupying the same locus of the pair of homologue chromosomes have the recessive value. Whether an allele is dominant or recessive is determined by a *dominance mechanism*. In nature, the dominance mechanism is determined by and also evolves with the environment.

Diploidy and dominance mechanisms have been borrowed and integrated into DGAs to enhance their performance in dynamic environments. DGAs have a framework similar to traditional GAs with two major differences. The first one lies in the representation and evaluation scheme. For DGAs the genotype and phenotype of an individual are separated as shown in Fig. 1. In order to evaluate an individual we need first map the diploid genotype (a pair of chromosomes) into a haploid phenotype according to some dominance mechanism. Then, the phenotype is evaluated according to the external environment and the fitness of the individual is obtained. For di-allelic representation where each gene has two alleles (0 or 1), the chromosomes in the genotype and the phenotype are binary strings. In this case, the black and white bars in Fig. 1 represents 0 and 1 respectively, vice versa.

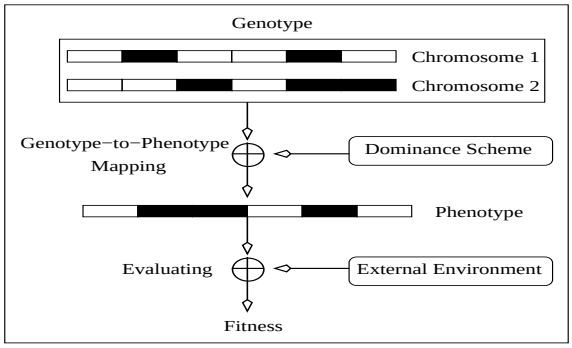

Fig. 1. Representation and evaluation of an individual for DGAs.

	0	o	1	i
0	0	0	0/1	0
o	0	0	1	0/1
1	0/1	1	1	1
i	0	0/1	1	1

(a)

	A	B	C	D
A	0	0	0	1
B	0	0	0	1
C	0	0	1	1
D	1	1	1	1

(b)

Fig. 2. Two dominance schemes: (a) Ng-Wong's and (b) Additive.

The second difference lies in the reproduction operations. For DGAs, crossover can be divided into two steps: exchange chromosomes and crossover. In the first step, two parents (each has a pair of chromosomes) exchange their chromosomes randomly to create two temporary offspring. Each offspring has one chromosome from each parent and hence the genotype materials from the parents are mixed and propagated to the offspring. In the second step, each offspring then undergoes the crossover operation within its own two chromosomes with a probability p_c, which is the same as in traditional GAs. For DGAs, mutation performs on each of the two genotype chromosomes of an individual independently with the same mutation probability p_m for each locus. Since mutation operates on the genotype, a mutation of a gene may or may not change the phenotype of an individual.

B. Dominance Schemes

As in biology, the dominance scheme controls how genes are expressed in the phenotype and plays a key role in the performance of DGAs. Hence, the research on DGAs has mainly focused on developing proper dominance schemes. There are several dominance schemes developed in the literature [4], [6], [10].

In [8], Ng and Wong proposed a diploid representation with simple dominance change for DOPs. They used four genotype alleles: dominant 1 and 0, and recessive i and o. The dominant allele is always expressed in the phenotype. If contention exists between two dominant or two recessive alleles, one is randomly chosen to be expressed. The occurrence of "1i" or "0o" is prohibited. If it occurs the recessive gene is promoted to be dominant. The genotype-to-phenotype mapping is shown in Fig. 2(a), where "0/1" means an equal probability of either 0 or 1. Ng and Wong also incorporated a dominance change scheme when the fitness of an individual drops by a preset percentage (20%) between successive evaluation cycles. Dominance change is achieved by inverting the dominance values of all allele-pairs such that "11" becomes "ii", "00" becomes "oo", "1o" becomes "i0", and vice versa. The genotype-to-phenotype mapping keeps unchanged.

In [9], Ryan proposed an *additive dominance* scheme, where genotype alleles are represented by ordered values that are combined using a pseudo-arithmetic to determine the phenotype allele. Ryan used four genotypic alleles A, B, C, and D, and allocated the values of 2, 3, 7, and 9 to them respectively. An addition is performed on the values allocated with the two genotypic alleles for each gene locus. If the result is greater than the threshold 10, the phenotypic allele becomes 1; otherwise, it becomes 0. The resulting dominance map is shown in Fig. 2(b).

In [7], Lewis *et. al* extended Ryan's scheme by adding a dominance change mechanism where genotypic alleles are demoted or promoted by one grade. For example, demoting a "B" makes it a "A" whereas promoting it makes it a "C". Allele "A" cannot be demoted and "D" cannot be promoted. As in the Ng-Wong approach, dominance change occurs when the fitness of a population member drops by a threshold percentage (20%) between successive evaluation cycles. If this happens, for each gene locus one of the two genotypic alleles is randomly chosen and the following procedure is performed: If the phenotypic allele at this locus is 1, then demote the chosen genotypic allele, unless it is an "A"; otherwise, promote the chosen genotypic allele, unless it is a "D". Lewis *et. al.* [7] compared the Ng-Wong and additive dominance schemes with or without dominance change on a dynamic knapsack problem and concluded that some form of dominance change is essential for DGAs to address DOPs.

III. Dominance Learning Scheme

As mentioned before, the dominance scheme affects DGA's performance significantly. This is also true in biology. However, in biology the dominance mechanism may change with the environment under the control of some type of enzyme. Modelling biological mechanism here may be beneficial. That is, it may be beneficial for DGAs if the dominance mechanism can be changed according to the current dynamic environment. This inspiration leads to the *dominance learning mechanism* proposed in this paper and the corresponding DGA is called *dominance learning DGA*, *DLDGA* in short.

DLDGA does not use a strict genotype-to-phenotype mapping scheme as shown in Fig. 3. Instead, we can define a *dominance probability vector* where each element is a *dominance probability* that represents the probability a genotypic allele can be expressed in the phenotype in the corresponding locus. The dominance probability vector is evolved with the dynamic environment by a learning scheme. Below we

describe the dominance learning mechanism regarding di-allelic encoding since this paper uses di-allelic encoding.

For di-allelic encoding, without loss of generality, we can define the dominance probability vector in terms of expressing allele 1 in the phenotype. That is, the dominance probability vector at generation t can be defined as $\vec{p}_d(t) = \{p_d(i,t), \cdots, p_d(l,t)\}$ where l is the encoding length. Here $p_d(i,t)$ denotes the probability that allele 1 will dominate allele 0 and appear in locus i of the phenotype of an individual if the two genotypic alleles of the individual do not agree in locus i. With this definition, the dominance scheme works as follows. Let $\vec{C}_1(t) = \{C_1(1,t), \cdots, C_1(l,t)\}$ and $\vec{C}_2(t) = \{C_2(1,t), \cdots, C_2(l,t)\}$ be the two chromosomes in the genotype of an individual in the population at time t, then $\vec{C}_1(t)$ and $\vec{C}_2(t)$ are mapped to the phenotype $\vec{P}(t) = \{P(1,t), \cdots, P(l,t)\}$ of the individual as follows:

$$P(i,t) = \begin{cases} 1, & C_1(i,t) = C_2(i,t) = 1 \\ 0, & C_1(i,t) = C_2(i,t) = 0 \\ 1, & C_1(i,t) \neq C_2(i,t) \ \& \ r < p_d(i,t) \\ 0, & C_1(i,t) \neq C_2(i,t) \ \& \ r >= p_d(i,t), \end{cases} \quad (1)$$

where $r = rand(0.0, 1.0)$ is a random number.

With respect to evolving the dominance probability vector we can use the learning scheme similar to that used in the Population-Based Incremental Learning (PBIL) algorithm, which was proposed by Baluja [1]. This learning scheme is described as follows. Originally, the dominance probability vector starts from the *neutral dominance probability vector* that has a value of 0.5 for each locus. That is, $\vec{p}_d(0) = \vec{0.5}$. This means the probability of expressing a 1 or 0 in the phenotype on each locus is equal when two genotypic alleles of a locus are different. Then, the dominance probability vector is updated every generation according to the current population. As in PBIL, the best individual(s) in the population can be extracted to learn $\vec{p}_d$ from. In this paper, only the best individual is used. Let $\vec{P}_B(t) = \{P_B(1,t), \cdots, P_B(l,t)\}$ denote the phenotype of the best individual at generation t. Then, $\vec{p}_d$ is learnt toward $\vec{P}_B(t)$ as follows:

$$p_d(i, t+1) := p_d(i,t) * (1-\alpha) + \alpha * P_B(i,t), \quad (2)$$

where $i \in \{1, \cdots, l\}$ and α is the learning rate.

From the above description, it can be seen that in DLDGA the dominance scheme is updated toward the current environment. This bias is expected to adapt DLDGA to the changing environment more efficiently than those dominance schemes that are unbiased to the current environment. For example, as seen from Fig. 3, both the Ng-Wong and additive schemes are unbiased since the total probability of creating a phenotypic allele 0 or 1 is exactly 0.5.

IV. Experimental Study

In this section, we present our experimental study of comparing DLDGA with two DGAs described in Section 2. The first, denoted *NWDGA*, is the DGA with the Ng-Wong dominance scheme with dominance change. The second DGA is the modified additive dominance scheme with dominance change, denoted *AddDGA* in this paper. For NWDGA and AddDGA, whenever the environment changes, the dominance change scheme in NWDGA and AddDGA starts to work.

A. Dynamic Test Environments

In this paper we use the DOP generator proposed in [11], [12] to construct dynamic test environments. This generator can construct DOPs from any binary-encoded stationary function $f(\vec{x})$ ($\vec{x} \in \{0,1\}^l$ and l is the encoding length) as follows. Suppose the environment changes every τ generations. For each environment k, an XORing mask $\vec{M}(k)$ is incrementally generated as follows:

$$\vec{M}(k) = \vec{M}(k-1) \oplus \vec{T}(k), \quad (3)$$

where "$\oplus$" is a bitwise exclusive-or (XOR) operator and $\vec{T}(k)$ is an intermediate binary template for environment k. $\vec{T}(k)$ is created with $\rho \times l$ ($\rho \in (0.0, 1.0]$) random loci set to 1 while the remaining loci set to 0. For the first initial environment $k = 1$, $\vec{M}(1)$ is set to a zero vector.

The evaluation of an individual at generation t is calculated as follows:

$$f(\vec{x}, t) = f(\vec{x} \oplus \vec{M}(k)), \quad (4)$$

where $k = \lceil t/\tau \rceil$ is the index of the environment at time t. With this XOR generator, τ and ρ control the speed and severity of environmental changes respectively. Smaller τ means faster changes while bigger ρ means severer changes.

Three 100-bit binary functions are selected as base functions to construct DOPs. The first is the well-known *OneMax* function that aims to maximize the number of ones in a binary string. The second one is a variant of Forrest and Mitchell's *Royal Road* function [5], which consists of 25 contiguous 4-bit building blocks (BBs). Each BB contributes 4 to the total fitness if all bits inside the BB have the allele of one; otherwise, it contributes 0. The third function also consists of 25 contiguous 4-bit BBs. Each BB for the third function is fully *deceptive* and is defined as below: it contributes 4 to the total fitness if all bits inside the BB have the allele of one; otherwise, it contributes 3 minus the number of ones inside the BB. All the three functions have an optimum fitness of 100.

Dynamic environments are constructed from each of the three base functions using the XOR DOP generator. For each dynamic environment, the landscape is periodically changed every τ generations during the run of a GA. In order to compare the performance of DGAs in different dynamic environments, τ is set to 10, 50 and 100 and ρ is set to 0.1, 0.2, 0.5, and 1.0 respectively. Hence, a series of 12 DOPs are constructed from each stationary function.

B. Experimental Settings

Experimental study was carried out to compare DLDGA, NWDGA and AddDGA on the above constructed dynamic environments. For all DGAs, parameter settings are as follows: generational, uniform crossover with the crossover probability $p_c = 0.6$, bit flip mutation with probability

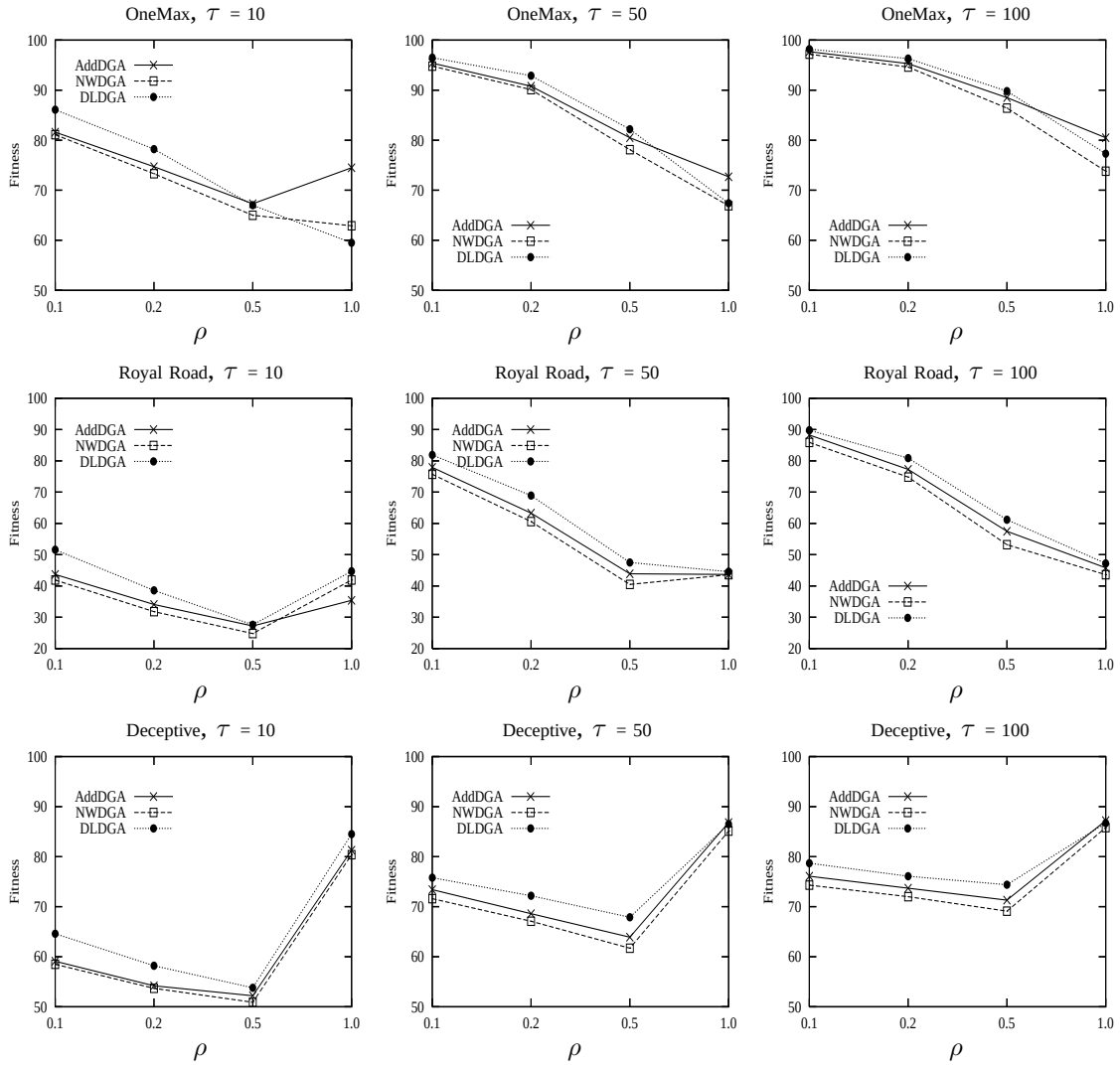

Fig. 3. Experimental results of DGAs on dynamic optimization problems.

$p_m = 0.01$, standard tournament selection with tournament size of 2, and elitism of size 1. The population size n is set to 100. For DLDGA, the learning rate α is set to 0.5.

For each experiment of a DGA on a dynamic problem, 50 independent runs were executed with the same set of random seeds. For each run 50 environmental changes were allowed, which are equivalent to 500, 2500 and 5000 generations for $\tau = 10$, 50 and 100 respectively. For each run the best-of-generation fitness was recorded every generation. The overall performance of an algorithm on a problem is defined as follows:

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

where $G = 50 * \tau$ is the total number of generations for a run, $N = 50$ is the number of runs, and $F_{BOG_{ij}}$ is the best-of-generation fitness of generation i of run j. The off-line performance $\overline{F}_{BOG}$ is the best-of-generation fitness averaged over 50 runs and then over the data gathering period.

C. Experimental Results and Analysis

The experimental results of comparing investigated DGAs on the constructed dynamic test problems are plotted in Fig. 3. The corresponding statistical results of comparing DGAs by one-tailed t-test with 98 degrees of freedom at a 0.05 level of significance are given in Table 1. In Table 1, the t-test result regarding $Alg.\ 1 - Alg.\ 2$ is shown as "$s+$" or "$s-$" when $Alg.\ 1$ is significantly better than or significantly worse than $Alg.\ 2$ respectively. In order to better understand the performance of DGAs in dynamic environments, their dynamic behaviour regarding best-of-generation fitness against generations on dynamic OneMax problems with $\tau = 50$ and $\rho = 0.2$ and $\rho = 1.0$ is plotted in Fig. 4, where the first 10 environmental changes are shown and the data were averaged over 50 runs. From the tables and figures several results can be observed.

First, DLDGA significantly outperforms both NWDGA and AddDGA on almost all DOPs except a few with $\rho = 1.0$, see the t-test results in Table 1. This result validates the

TABLE I

THE t-TEST RESULTS OF COMPARING DGAS ON DYNAMIC OPTIMIZATION PROBLEMS.

t-test Result	$OneMax$				$Royal\ Road$				$Deceptive$			
$\tau = 10,\ \rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
$DLDGA - AddDGA$	$s+$	$s+$	$s-$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$DLDGA - NWDGA$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$AddDGA - NWDGA$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$
$\tau = 50,\ \rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
$DLDGA - AddDGA$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$
$DLDGA - NWDGA$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$AddDGA - NWDGA$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$\tau = 100,\ \rho \Rightarrow$	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0	0.1	0.2	0.5	1.0
$DLDGA - AddDGA$	$s+$	$s+$	$s+$	$s-$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s-$
$DLDGA - NWDGA$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$
$AddDGA - NWDGA$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$	$s+$

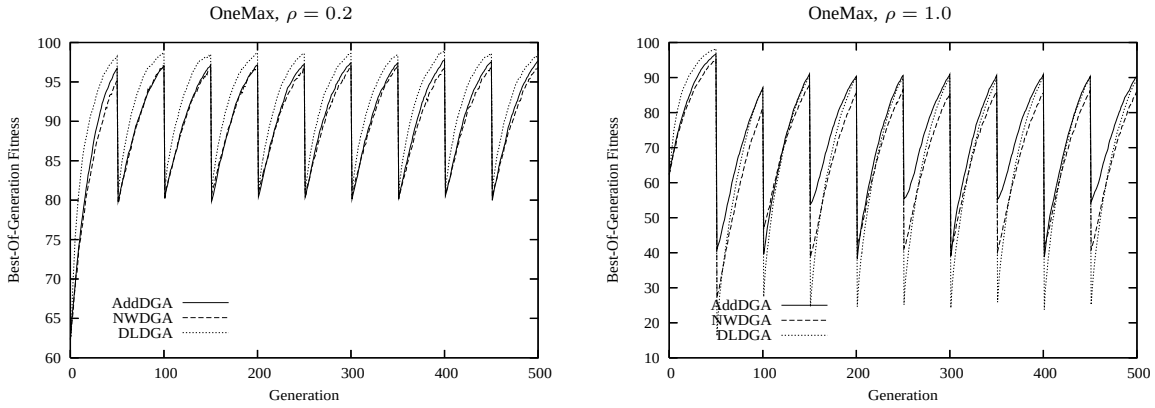

Fig. 4. Dynamic behaviour of DGAs on $OneMax$ with $\tau = 50$ and $\rho = 0.2$ and 1.0.

efficiency of introducing the dominance learning scheme over fixed ones into DGAs for DOPs. The effect of the dominance learning scheme can be more clearly observed from the dynamic behaviour of DGAs as shown in the left column of Fig. 4: every time the environment changes DLDGA can more rapidly climbs to a higher fitness level than NWDGA and AddDGA can do.

When $\rho = 1.0$, i.e., the environment oscillates between two opposite fitness landscape, DLDGA is beaten by AddDGA on some DOPs, especially on the OneMax problem. The underlying reason can be observed from examining the dynamic behaviour of DGAs. In the right column of Fig. 4, it can be seen that each time the environment changes AddDGA achieves much less fitness dropping than DLDGA does. This happens because the dominance change scheme in AddDGA is in fact more directly oriented toward extreme environmental changes as with the oscillating case when $\rho = 1.0$. In this case, though DLDGA can still climb to a fitness height comparable to or even higher than the height AddDGA reaches at the end of each environmental period, the overall average gives AddDGA a lead over DLDGA.

Second, comparing AddDGA and NWDGA, it can be seen that AddDGA outperforms NWDGA on almost all DOPs, see the t-test results in Table 1. This is an interesting result and seems difficult to understand at the first glance since both use similar dominance change schemes, as described in Section II. However, scrutinizing their dominance schemes in Fig. 2 may dig out the deeply hidden secret. Comparing Fig. 2(a) and Fig. 2(b), it can be seen that in the Ng-Wong scheme there are four "0/1" uncertain values in the map. In other words, in the Ng-Wong scheme when "1" meets "0" or "i" meets "o" in the genotype, the phenotypic allele is random. This puts NWDGA in a disadvantageous ground over AddDGA. This is because for each given environment the allele for each locus in the optimal solution is deterministic for the test DOPs and hence having some phenotypic alleles randomly wandering between 0 and 1 may be disadvantageous.

Third, when observing the performance of DGAs on the deceptive DOPs, it can be seen that DGAs perform much better when $\rho = 1.0$ than when ρ is set to other values. This is because in the deceptive function there is a strong deceptive sub-optimal schema in its building block, which is complementary to the global optimal schema. When $\rho = 1.0$ the individuals of the population will mainly switch between the global optimal and the sub-optimum.

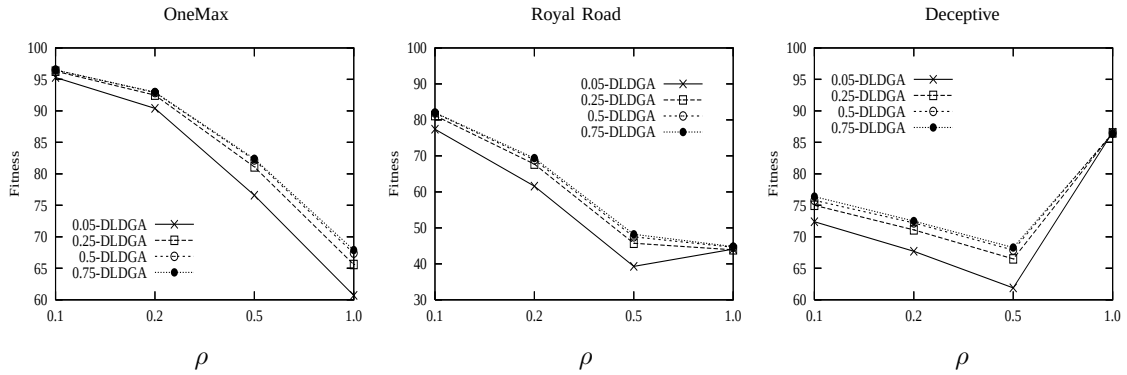

Fig. 5. Experimental results on the effect of α on DLDGAs on DOPs with $\tau = 50$.

D. Sensitivity Analysis of the Learning Rate

It is intuitive that the learning rate α has an important effect on DLDGA's performance in dynamic environments. In order to investigate this effect, we further carried out experiments on DLDGA with different settings, 0.05, 0.25, and 0.75, for α on the dynamic test problems. The experimental design is the same as previous one for DLDGAs except the learning rate. The experimental results are plotted in Fig. 5 where the data were obtained and averaged over 50 runs and the DLDGA with learning rate α is marked as α-DLDGA accordingly.

From Fig. 5, it can be seen that the learning rate α does affect DLDGA's performance on the DOPs. When the value of α rises from 0.05 to 0.25, the performance of DLDGA improves significantly on most DOPs. When α is raised to 0.5, DLDGA's performance is further improved but with much less magnitude. When the value of α is further raised to 0.75, the performance of DLDGA is only slightly better on the dynamic test problems.

V. Conclusions

This paper proposes a dominance learning scheme that uses a dominance probability vector to map the genotype to phenotype of individuals for DGAs to address DOPs. This dominance probability vector is learnt adaptively toward the current environment and hence can adapt the DLDGA more efficiently in the changing environment. Experiments were carried out based on a series of systematically constructed DOPs to compare the proposed dominance learning scheme with two other dominance change schemes for DGAs. From the experimental results and relevant analysis, several conclusions can be drawn on the dynamic test environments.

First, the proposed dominance learning scheme is efficient to improve the performance of DGAs in dynamic environments. It outperforms the other two investigated dominance change schemes.

Second, the parameter sensitivity analysis shows that the learning rate α has a significant effect on the proposed dominance learning scheme and setting α in the range of $[0.25, 0.5]$ seems a good choice for DLDGA.

Third, the additive dominance scheme is significantly better than the Ng-Wong scheme for DGAs on the dynamic test environments. The existence of uncertainty in the dominance mapping in the Ng-Wong scheme seems giving it a disadvantage over the additive dominance scheme.

In this paper, we have carried out some preliminary experiments comparing the proposed dominance learning scheme for DGAs over two existing dominance change schemes. Comparing it with other advanced dominance schemes, such as the one in [10], is now under investigation. It is also an interesting future work to compare it with some advanced explicit memory schemes for GAs for DOPs.

References

[1] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. *Technical Report CMU-CS-94-163*, Carnegie Mellon University, USA, 1994.

[2] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. *Proc. of the 1999 Congress on Evolutionary Computation*, vol. 3, pp. 1875-1882, 1999.

[3] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

[4] E. Collingwood, D. Corne, and P. Ross. Useful diversity via multiploidy. *Proc. of the IEEE 1996 Int. Conf. on Evolutionary Computation*, pp. 810-813, 1996.

[5] S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hyperthesis. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2*, pp. 109-126, 1992.

[6] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, pp. 59-68, 1987.

[7] E. H. J. Lewis and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. *PPSN V*, pp. 139-148, 1998.

[8] K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimisation. *Proc. of the 6th Int. Conf. on Genetic Algorithms*, pp. 159-166, 1995.

[9] C. Ryan. The degree of oneness. *Proc. of the 1994 ECAI Workshop on Genetic Algorithms*, 1994.

[10] A. Ş. Uyar and A. E. Harmanci. A new population based adaptive dominance change mechanism for diploid genetic algorithms in dynamic environments. *Soft Computing*, 9(11): 803-814, 2005.

[11] S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithm. *Proc. of the 2003 Congress on Evolutionary Computation*, vol. 3, pp. 2246-2253, 2003.

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