Non-Stationary Problem Optimization Using the Primal-Dual Genetic Algorithm

Shengxiang Yang
Department of Computer Science
University of Leicester
University Road, Leicester LE1 7RH, United Kingdom
Email: s.yang@mcs.le.ac.uk

Abstract- Genetic algorithms (GAs) have been widely used for stationary optimization problems where the fitness landscape does not change during the computation. However, the environments of real world problems may change over time, which puts forward serious challenge to traditional GAs. In this paper, we introduce the application of a new variation of GA called the Primal-Dual Genetic Algorithm (PDGA) for problem optimization in non-stationary environments. Inspired by the complementarity and dominance mechanisms in nature, PDGA operates on a pair of chromosomes that are primal-dual to each other in the sense of maximum distance in genotype in a given distance space. This paper investigates an important aspect of PDGA, its adaptability to dynamic environments. A set of dynamic problems are generated from a set of stationary benchmark problems using a dynamic problem generating technique proposed in this paper. Experimental study over these dynamic problems suggests that PDGA can solve complex dynamic problems more efficiently than traditional GA and a peer GA, the Dual Genetic Algorithm. The experimental results show that PDGA has strong viability and robustness in dynamic environments.

1 Introduction

As a class of evolutionary algorithms that make use of principles of natural selection and population genetics, genetic algorithms (GAs) are widely and often used for solving *stationary* optimization problems where the fitness landscape or objective function does not change during the course of computation [Goldberg89a]. Once a satisfactory solution is found or certain termination condition is satisfied, the search stops. However, the environments of real world optimization problems may fluctuate or change sharply. For these non-stationary problems the objective function changes over the course of optimization, which requires that the GA must be able to track the trajectory of the moving optimal point(s) in the search space. This presents serious challenge to traditional simple GA (SGA) since they cannot adapt to changing functionality once converged.

Usually the dynamic environment requires GAs to maintain sufficient diversity for a continuous adaptation to the changing landscape. To improve GA's performance in dynamic environments, researchers have applied the *diploidy* and *dominance* mechanisms that broadly exist in nature [GS87], [NW95]. In nature, most organisms have a diploid genotype, i.e., a set of double-stranded chromosomes. When the double-stranded chromosomes are exposed to the environment of the organism, dominance mechanism takes effect by expressing dominant genes (a gene is a segment of DNA) while repressing recessive genes. Introducing diploid encoding into GAs has achieved some success, especially in non-stationary environments, since the additional information stored in the genotype provides a latent source of *diversity* in the population and allows the population to respond more quickly to the fitness changes [LHR98].

Inspired by the complementarity and dominance mechanisms in nature a new genetic algorithm called primal-dual genetic algorithm (PDGA) has been proposed [Yang03]. Within PDGA, each chromosome is defined a dual chromosome that is of maximum distance in genotype to it in a given distance space, e.g., the Hamming distance space. During the running of PDGA before iterating into next generation a set of low fit individuals is selected to evaluate their duals and give those dual chromosomes that are superior chances to be expressed into the next generation. The primal-dual mapping between primal-dual chromosomes improves PDGA's exploration capacity in the search space and thus its searching efficiency.

In this paper, we investigate the application of PDGA for non-stationary problem optimization. A new selection scheme that can adaptively adjust the number of primal chromosomes to be selected for dual evaluations is proposed for PDGA instead of the deterministic scheme used in [Yang03]. A dynamic problem generating technique is proposed that can generate a set of dynamic problems from a given stationary problem. Using this dynamic problem generating technique a set of dynamic problems is generated from a set of stationary benchmark problems and based on these generated dynamic problems we compare the performance of PDGA over SGA and a peer GA, Collard and co-workers' Dual Genetic Algorithm (DGA) [CA94], [CE96].

The rest of this paper is organized as follows. The next section describes the framework of PDGA, the new adaptive selecting scheme for dual evaluations, and Collard and co-workers' DGA in detail. Section 3 presents the new dynamic problem generating technique that is used in this study to create dynamic problems from a set of stationary test problems. Section 4 provides the experimental results and analysis of investigated GAs on the test environments. Finally Section 5 concludes this paper.

2 Primal-Dual Genetic Algorithm

2.1 Framework of PDGA

For the convenience of descriptions, we first introduce some definitions here. A chromosome explicitly recorded in the

0-7803-7804-0 /03/$17.00 © 2003 IEEE

population is called a *primal chromosome*. Given a distance space, the chromosome that has the maximum distance to a primal chromosome x is called its *dual chromosome*, denoted by $x' = dual(x)$ where $dual(\cdot)$ is the *primal-dual mapping* function.

In this paper we will deal with binary-encoded GAs and naturally use the Hamming distance (the number of locations at which corresponding bits of two chromosome differ) as the primal-dual mapping function. A pair of chromosomes is then said to be primal-dual to each other if their Hamming distance is the maximum in the search space. In other words, given a chromosome $x = (x_1, x_2, ..., x_L) \in I = \{0,1\}^L$ of fixed length L, its dual is its complement, i.e., $x' = dual(x) = \bar{x} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_L) \in I$ where $\bar{x}_i = 1 - x_i$ $(i = 1, ..., L)$.

For a pair of primal-dual chromosomes, if they have different fitness the chromosome with higher fitness is called a *superior chromosome* while the other an *inferior chromosome*; otherwise, if the pair have equal fitness, they are called *tie chromosomes* or said to form a *tie pair*. A primal-dual mapping operation or dual evaluation is called *valid* if the obtained dual chromosome is superior to the primal chromosome; otherwise, it is called *invalid*. Valid primal-dual mappings or dual evaluations are expected to be beneficial to GA's performance. This is the fundamental working principle of PDGA.

With above definitions, the framework of PDGA is shown in Figure 1, where N, P_c, P_m are the population size, crossover probability and mutation probability respectively, and $f(x)$ denotes the fitness of an individual x. Within PDGA, when an intermediate population $P'(t)$ has just been created and evaluated and before the next generation starts, a set $D(t)$ of individuals are selected from $P'(t)$ to evaluate their duals. For an individual $x \in D(t)$, if its dual x' is evaluated to be fitter, x is replaced with x'; otherwise, x will survive into the next generation. That is, only valid primal-dual mappings take effect to give superior dual chromosomes chances to be expressed into the next generation. This is similar to the dominance mechanism in nature except that now it works at the chromosome level. Although inspired by the complementary mechanism in nature, physically within PDGA only the primal chromosomes need being recorded in the population. That is, the encoding is based on a single-stranded chromosome instead of double-stranded as in DNA molecule. Hence, PDGA can be called *pseudo-diploid* or called working on a *pseudo-pair* of primal-dual chromosomes.

Procedure PDGA:

begin
  parameterize(N, P_c, P_m);
  $t := 0$;
  initializePopulation($P(0)$);
  evaluatePopulation($P(0)$);
  $D(0)$:= selectForDualEvaluation($P(0)$);
  for each individual x in $D(0)$ **do** // *evaluate* $D(0)$
    evaluateDualChromosome(x'); // $x' = dual(x)$
    if $f(x') > f(x)$ **then** replace x in $P(0)$ with x';
  endfor;
  repeat
    $P'(t)$:= selectForReproduction($P(t)$);
    recombine($P'(t)$);
    mutate($P'(t)$);
    evaluatePopulation($P'(t)$);
    $D(t)$:= selectForDualEvaluation($P'(t)$);
    for each individual x in $D(t)$ **do** // *evaluate* $D(t)$
      evaluateDualChromosome(x'); // $x' = dual(x)$
      if $f(x') > f(x)$ **then** replace x in $P'(t)$ with x';
    endfor;
    $t := t + 1$;
  until *terminated* **= true**; // *e.g.*, $t > t_{max}$
end;

Figure 1: Pseudocode for PDGA. In Hamming distance space, $x' = dual(x) = \bar{x}$.

2.2 Adaptive Selection Scheme for Dual Evaluation

From above discussions, it is clear that the scheme of selecting primal chromosomes from $P'(t)$ to form the set $D(t)$ for dual chromosome evaluation should try to maximize valid primal-dual mappings, i.e., selecting as many inferior primal chromosomes in the population as possible. Hence, the selection scheme should concern which primal chromosomes and how many primal chromosomes in $P'(t)$ should be selected.

Since only valid primal-dual mappings take effect and performing primal-dual mapping on chromosomes with low fitness is more likely to be valid, we can select primal chromosomes with low fitness from $P'(t)$ to form the set $D(t)$. Let $n_d(t)$ denote the actual number of primal chromosomes selected from $P'(t)$ into $D(t)$ for dual evaluation at generation t, i.e., $n_d(t) = |D(t)|$. For each generation t, we can select $n_d(t)$ least fit primal chromosomes from $P'(t)$ for dual evaluations. Now what is left is how to decide the value of $n_d(t)$.

Both Holland's [Holland75] and Stephens and Waelbroeck's [SW99] schema theorems indicate that schemas or strings with less than average fitness or average effective fitness receive an exponentially decreasing number of trials over time. Let $n_{inf}(t)$ denote the actual number of inferior primal chromosomes in the population $P'(t)$. The schema theorems indicate that $n_{inf}(t)$ will decrease at an exponential rate since they usually have less than average fitness or average effective fitness. This is verified by our experiments in [Yang03], which shows that $n_{inf}(t)$ decreases approximately exponentially over time t to 0 or an approximately stable value. Ideally, the value of $n_d(t)$ should be equal to $n_{inf}(t)$. However, it is difficult to achieve this since $n_{inf}(t)$ is unknown in advance.

In [Yang03] we have used a simple deterministic scheme to decrease $n_d(t)$ exponentially as follows:

$$n_d(t) = \begin{cases} \lceil \alpha * N \rceil, & \text{if } t = 0 \\ max\{\lceil \beta * n_d(t-1) \rceil, n_m\}, & \text{if } t > 0 \end{cases} \quad (1)$$

where α, $\beta \in (0,1)$ control the initial value and the decreasing speed of $n_d(t)$ respectively, $n_m \in (0,N)$ is the minimal number of primal chromosomes to be selected for dual evaluation, and $\lceil y \rceil$ is the ceiling function returning the minimum integer that is not less than y. With this primal chromosome selection scheme after several generations the value of $n_d(t)$ will stay constant at n_m.

In this paper we propose an adaptive scheme to decide the value of $n_d(t)$. Let $n_v(t-1)$ denote the actual number of valid dual evaluations carried out during generation $t-1$ then $n_d(t)$ is calculated as follows:

$$sign(t) = \begin{cases} 1, & \text{if } n_v(t-1)/n_d(t-1) < \delta \\ 0, & \text{if } n_v(t-1)/n_d(t-1) = \delta \\ -1, & \text{if } n_v(t-1)/n_d(t-1) > \delta \end{cases} \quad (2)$$

$$n_d(t) = \begin{cases} \lceil \alpha * N \rceil, & \text{if } t = 0 \\ min\{n_M, max\{\lceil \beta^{sign(t)} * n_d(t-1) \rceil, n_m\}\}, & \text{if } t > 0 \end{cases} \quad (3)$$

where in Equation (2), $\delta \in (0,1)$ is a preset constant (e.g., 0.9) that acts as the threshold of $n_d(t)$'s changing mode: decreasing, increasing, or neither. In Equation (3), $\alpha \in (0,1)$ has the same meaning as in Equation (1), $\beta \in (0,1)$ controls the decreasing or increasing speed of $n_d(t)$, and n_M, $n_m \in (0,N)$ are preset maximal and minimal number of primal chromosomes to be selected from $P'(t)$ for dual evaluations respectively. With this adaptive primal chromosome selection scheme, now $n_d(t)$ can decrease, keep unchanged, or increase adaptively depending on whether the measured ratio of valid dual evaluations to total dual evaluations during generation $t-1$, i.e., $n_v(t-1)/n_d(t-1)$, is less than, equal to, or greater than the preset threshold δ respectively. This selection scheme aims to give PDGA even stronger adaptability, especially in dynamic environments where the value of $n_i(t)$ may vary with time.

2.3 PDGA vs. Dual Genetic Algorithm

Collard and his co-workers proposed a genetic algorithm, called Dual Genetic Algorithm (DGA) [CA94], [CE96]. The initial aim of DGA is to improve the performance of a GA by adding one single meta-bit in front of the regular bits. This meta-bit alters the phenotype of the overall chromosome. If the meta-bit is activated ("1") all regular bits are translated to their complement for fitness evaluation; otherwise, they keep their original alleles for fitness evaluation. Consequently, there may exist complementary individuals in the population that represent the same phenotype though they have fundamentally different genotype. The added meta-bit undergoes the same genetic operations within DGA as other regular bits do.

Both PDGA and DGA are inspired by the complementary mechanism in nature, such as DNA's duplex structure. In DGA mutating the meta-bit enables an individual to make a long jump to its complement in the search space while in PDGA when a primal chromosome is selected into the set $D(t)$, it will get the chance to jump to its dual. However, PDGA and DGA do have different properties. The main difference lies in that in DGA the jumping between complementary chromosomes is driven by mutation and hence by chance. It uses no dominance mechanism and is blind in the sense of applying complementary mechanism. PDGA is dominance-based using fitness as its dominance mechanism that works at the chromosome level. This is reflected in selecting low fit chromosomes for dual evaluation and only replacing inferior primal ones with their superior duals.

3 The Test Suite

A set of well studied stationary problems, forming a range of difficulty levels for GAs, is selected as the test set to compare the performance between PDGA, SGA and DGA. Dynamic test problems are constructed from these stationary problems by a dynamic problem generating technique introduced in this paper.

3.1 Stationary Test Problems

1. **One-Max Problem**:

This problem was studied by Ackley [Ackley87] and simply aims to maximize ones in a binary string. The fitness of a string is the number of ones it contains. A string length of 100 bits will be used for this study. And the unique optimum solution has a fitness of 100.

2. **Royal Road Function**:

This function is the same as Mitchell, Forrest and Holland's Royal Road function $R1$ [MFH92]. It is defined on a sixty-four bit string consisting of eight contiguous building blocks (BBs) of eight bits, each of which contributes $c_i = 8$ $(i = 1, ..., 8)$ to the total fitness if all of the eight bits are set to one. The fitness of a bit string x is computed by summing the coefficients c_i corresponding to each of the given BBs s_i, of which x is an instance (denoted by $x \in s_i$). That is, the royal road function is defined as follows:

$$f(x) = \sum_{i=1}^{i=8} c_i \delta_i(x)$$

where $\delta_i(x) = \{1, \text{if } x \in s_i; 0, \text{otherwise}\}$. This function has an optimum fitness of 64.

3. **Deceptive Function**:

Deceptive functions are a family of functions where there exist low-order BBs that do not combine to form higher-order BBs: instead they form BBs resulting in a deceptive solution that is sub-optimal itself or near a sub-optimal solution [Whitley91]. Deceptive functions are devised as difficult test functions for GAs.

Goldberg [Goldberg89b] devised an order-3 minimum fully deceptive problem as follows:

```
f(000) = 28   f(001) = 26
f(010) = 22   f(011) = 0
f(100) = 14   f(101) = 0
f(110) = 0    f(111) = 30
```

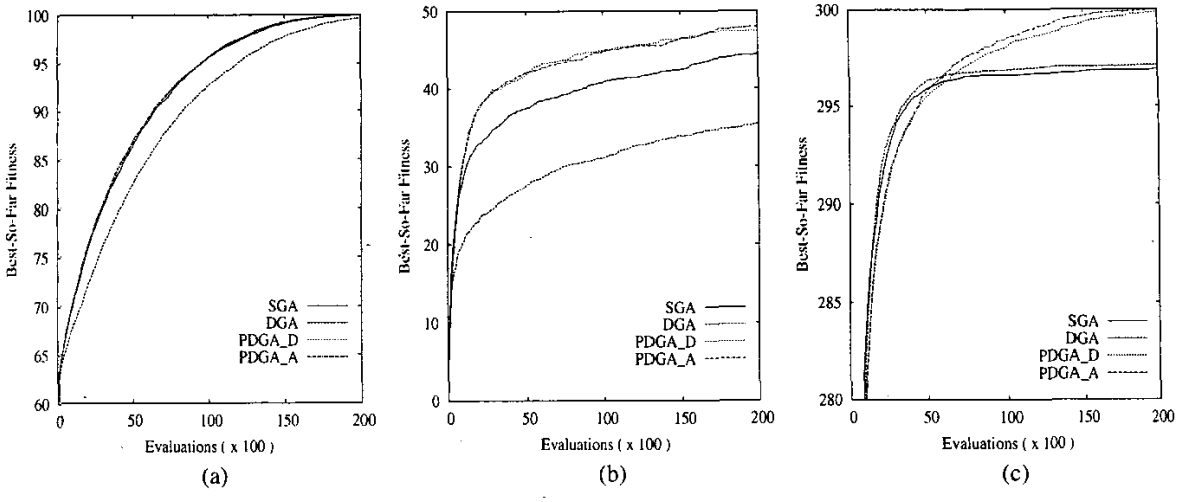

Figure 2: Experimental results with respect to best-so-far fitness against evaluations of GAs on stationary test problems: (a) One-Max, (b) Royal Road, and (c) Deceptive Function.

where all the order-1 and order-2 BBs are deceptive. In this study, we constructed a deceptive function consisting of 10 copies of the above deceptive subproblem. This function has an optimum fitness of 300.

3.2 Generating Dynamic Test Problems

In this paper, we introduce a new technique that generates a dynamic test problem from a given stationary problem. Given a binary-encoded stationary optimization problem $f(x)$, we can construct its dynamic version as follows. First, create a binary template T of the chromosome length, randomly or in a controlled way, periodically or not. Second, for each chromosome x in the population perform the operation $x \oplus T$ where $\oplus$ is the bitwise exclusive-or operator (i.e., $1 \oplus 1 = 0$, $1 \oplus 0 = 1$, $0 \oplus 0 = 0$). Suppose that the environment changes at generation t, then at generation $t+1$ we have $f(x, t+1) = f(x \oplus T, t)$. In this way, we can change the fitness landscape but still keep certain properties of the original fitness landscape, e.g., the total number of optima and fitness values of optima though their locations shifted. For example, if we apply a template $T = 11111$ to a 5-bit One-Max function, the original optimal point $x^* = 11111$ becomes the least fit point while original least fit point $x = 00000$ becomes the new optimal point in the changed landscape, but the optimal fitness value (i.e., 5) and the uniqueness of optimum keep invariant.

In this paper, we construct dynamic versions of above stationary problems in two ways. First, in the periodically randomly shifting version, every 200 generations the fitness landscape is randomly shifted with a randomly created template T that contains half ones and half zeros (assuming L is even). This case results in a moderate change in the environment in the sense of Hamming distance. An optimal point in the search space is shifted to some position, located half way between the original optimum and its dual. Second, in the periodically reversing version every 200 generations the fitness landscape is reversed with the template T containing all 1s, i.e., $f(x, t+200) = f(x \oplus T, t) = f(\bar{x}, t)$. This case brings in the extreme or maximum landscape change in the sense of Hamming distance, analogous to natural environment change between sunny daytime and dark night.

4 Experimental Study

Experiments were carried out to compare PDGA with traditional SGA and Collard and his co-workers' DGA. All the GAs were generational and used typical genetic operator and parameter settings: 1-point crossover with a fixed crossover probability $p_c = 0.6$, bit mutation with mutation probability $p_m = 0.001$, and fitness proportionate selection with the Stochastic Universal Sampling (SUS) [Baker87] and elitist model [DeJong75]. The population size N was set to 128 for all the GAs. PDGA-specific parameters were set as follows: for the deterministic selection scheme $\alpha = \beta = 0.5$ and $n_m = 1$; for the adaptive selection scheme $\alpha = \beta = 0.5$, $\delta = 0.9$, $n_M = N/2$ and $n_m = 1$. For each experiment of combining different GA and test problem, 100 independent runs were executed with the same 100 random seeds to generate initial populations. Each experimental result was averaged over the 100 runs.

4.1 Experimental Results on Stationary Problems

Preliminary experiments were carried out on the stationary version of the three test problems. For each run of different GAs on each problem, the best-so-far fitness was recorded every 100 evaluations. Here, only those primal chromosomes changed by crossover and mutation operations were evaluated and counted into the number of evaluations. With PDGA all dual evaluations, valid or not, were also counted into the number of evaluations. For each run the maximum allowable evaluations was set to 20000. The experimental results are shown in Figure 2.

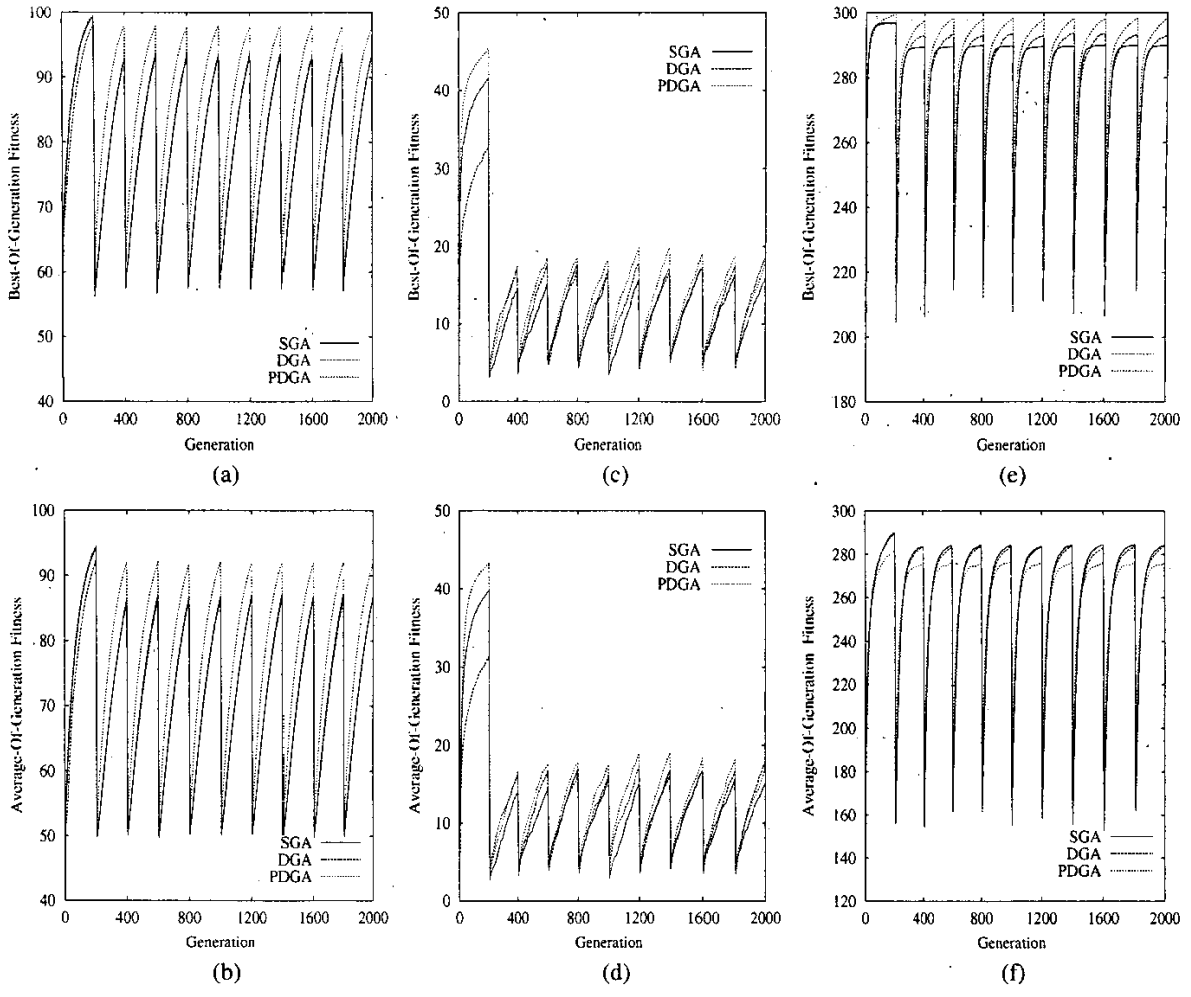

Figure 3: Experimental results with respect to best-of-generation fitness (*Top*) and average-of-generation fitness (*Bottom*) against generations of GAs on periodically randomly shifting dynamic test problems: One-Max (*a* & *b*), Royal Road (*c* & *d*), and Deceptive Function (*e* & *f*).

From Figure 2, it can be seen that in general PDGA performs better than SGA and DGA[1] and within PDGA the adaptive selection scheme marked by PDGA_A is better than the deterministic selection scheme marked by PDGA_D. For the following experiments on dynamic problems, only the adaptive selection scheme will be used within PDGA and the results are marked by PDGA instead of PDGA_A.

4.2 Experimental Results and Analysis on Periodically Randomly Shifting Dynamic Problems

For each run of different GAs on the periodically randomly shifting problems, the fitness function was randomly shifted every 200 generations[2] and the maximum allowable generations was set to 2000, which equals 10 periods of environment changes[3]. For each run the best-of-generation fitness and average-of-generation fitness were recorded every 5 generations. The experimental results are shown in Figure 3. From Figure 3 it can be seen that PDGA performs better than both SGA and DGA on the periodically randomly shifting dynamic problems.

On the One-Max problem (see Figure 3(a) and 3(b)), for the stationary period PDGA performs as well as SGA and both perform better than DGA (the same as the stationary situation in Figure 2(a)). However, for the dynamic periods PDGA outperforms both SGA and DGA. The valid primal-dual mappings within PDGA contribute to the faster growth speed of best-of-generation fitness as well as the average-of-

[1] See [Yang03] for the experimental result analysis relevant to this part.

[2] In the computer implementation of elitism for dynamic problems every time the fitness landscape changes the recorded elitist of previous period is discarded and reset to be the best individual of the changed generation.

[3] For the convenience of analyzing the experimental results, below in this paper we call the first period *stationary period* since the behavior of GAs on a dynamic problem during this period is the same as that on the relevant stationary problem. And the other 9 periods are called *dynamic periods*.

generation fitness of the population. On the contrast, with DGA due to the blindness in mutating the meta-bit it performs just the same as SGA and its performance curves almost overlap with SGA's. During dynamic periods the highest best-of-generation fitness and average-of-generation fitness reached by all the GAs are lower than those reached during the stationary period. This is because, due to the relative convergence of the last generation of previous period, the first generation of each dynamic period has a lower diversity than that of the stationary period.

On the Royal Road function (see Figure 3(c) and 3(d)), PDGA outperforms both SGA and DGA for all periods while SGA outperforms DGA for the stationary period (see Figure 2(b) for reference) and is beaten by DGA for dynamic periods. Now for the dynamic periods all the GAs perform much worse than they did for the stationary period. None of the GAs ever achieves three BBs (i.e., fitness value of 24) for the dynamic periods. And for the dynamic periods all the GAs perform worse on the Royal Road function than they did on the One-Max problem. The reason to these two observations lies in that the basic BBs in the Royal Road function are now of order-8 instead of order-1 as in the One-Max problem. The increased size of basic BBs makes it much harder for the GAs to search them and also enhances the "hitchhiking" effect [FM93]. Hitchhiking seriously decreases the diversity of those loci corresponding to BBs not found at the last generation of the stationary period. And the fitness landscape changing mode destroys almost all BBs found so far. Both sides together leave the GAs a very harsh situation to start from for all of the dynamic periods.

On the Deceptive Function (see Figure 3(e) and 3(f)), the situation seems a little different. With respect to the best-of-generation fitness PDGA outperforms both SGA and DGA while DGA outperforms SGA. On the contrast, the average-of-generation fitness with PDGA is less than that with SGA and DGA, especially at the late stage of each period. The reason lies in that after certain generations of each period the competition on each 3-bit subproblem is mainly between BBs "000" and "111". With PDGA valid primal-dual mappings, which convert strings with more "000" BBs to strings with more "111" BBs, work quite efficiently, pushing the best individual towards optimum faster than SGA and DGA. Meanwhile, these valid mappings give crossover more chances to create non-"000" or non-"111" BBs on certain loci and hence lower the average fitness of the population a little. With DGA for dynamic periods it seems now mutating the meta-bit helps pushing best-of-generation fitness toward optimum faster than SGA.

4.3 Experimental Results and Analysis on Periodically Reversing Dynamic Problems

The experimental setting for the periodically reversing version of problems was the same as that for above periodically randomly shifting version except that now for each run the fitness landscape was reversed every 200 generations with maximum Hamming distance instead of half Hamming distance. The experimental results are shown in Figure 4. From Figure 4 it can be seen that now GAs behave quite differently from what they did on the periodically randomly shifting problems and that PDGA now outperforms SGA and DGA with a much higher degree than it did on the periodically randomly shifting problems.

On the One-Max problem (see Figure 4(a) and 4(b)), after the stationary period PDGA keeps performing perfectly well with the best-of-generation fitness staying in the optimum over all dynamic periods while the average-of-generation fitness first drops a little when changes occur, then rises up quickly to near optimum value. This is because the complementary and dominance mechanisms embedded in PDGA work perfectly under the condition of extreme environment change. Whenever change occurs they together immediately replace the worst individuals in the changed landscape that are reversed from optimal individuals of previous period with their superior duals, resulting in new optimal individuals. With DGA the mechanism of mutating the meta-bit by probability now also works but with a slower speed than that of PDGA. As to SGA, without extra surviving mechanisms to deal with extreme environment change it now performs even worse than it did on the randomly shifting version of the One-Max problem. Comparing Figure 4(a & b) with Figure 3(a & b) makes this point clear.

On the Royal Road Function (see Figure 4(c) and 4(d)), PDGA outperforms both SGA and DGA while SGA and DGA beat each other interleavingly over the 10 periods. For PDGA whenever the fitness function changes the complementarity and dominance mechanisms in PDGA rapidly draw back the best individuals in the last generation of previous period (since the dual of a BB's dual is the BB itself). And then PDGA continues to evolve during the dynamic period. This results in both fitness measures getting better and better across dynamic periods with only slight dips for a short time just after each change happens. The scheme of mutating meta-bit in DGA has similar effect as the complementarity and dominance mechanisms in PDGA but with a slower speed. It is interesting to see that SGA behaves oscillatingly over different periods around two fitness levels: at a normal fitness level during odd periods (including the stationary period) and at a lower fitness level during even periods. The reason to this oscillating phenomenon is given as follows. When the first change occurs, SGA is left in a harsh environment as explained in Section 4.2 and from here it performs poorly during the whole period. When the second change happens, the fitness of the population is brought back to its normal level quite quickly due to the poor performance of SGA during previous period, which protects the BBs found in the stationary period quite well. This process continues for the following periods.

On the Deceptive Function, with respect to the best-of-generation fitness PDGA outperforms both SGA and DGA while DGA outperforms SGA (see Figure 4(e)). And for dynamic periods PDGA reaches a higher near-optimal best-of-generation fitness than it does for the stationary period while DGA reaches the same level as it does for the stationary period and SGA performs worse. With respect to the average-of-generation fitness (see Figure 4(f)), with PDGA

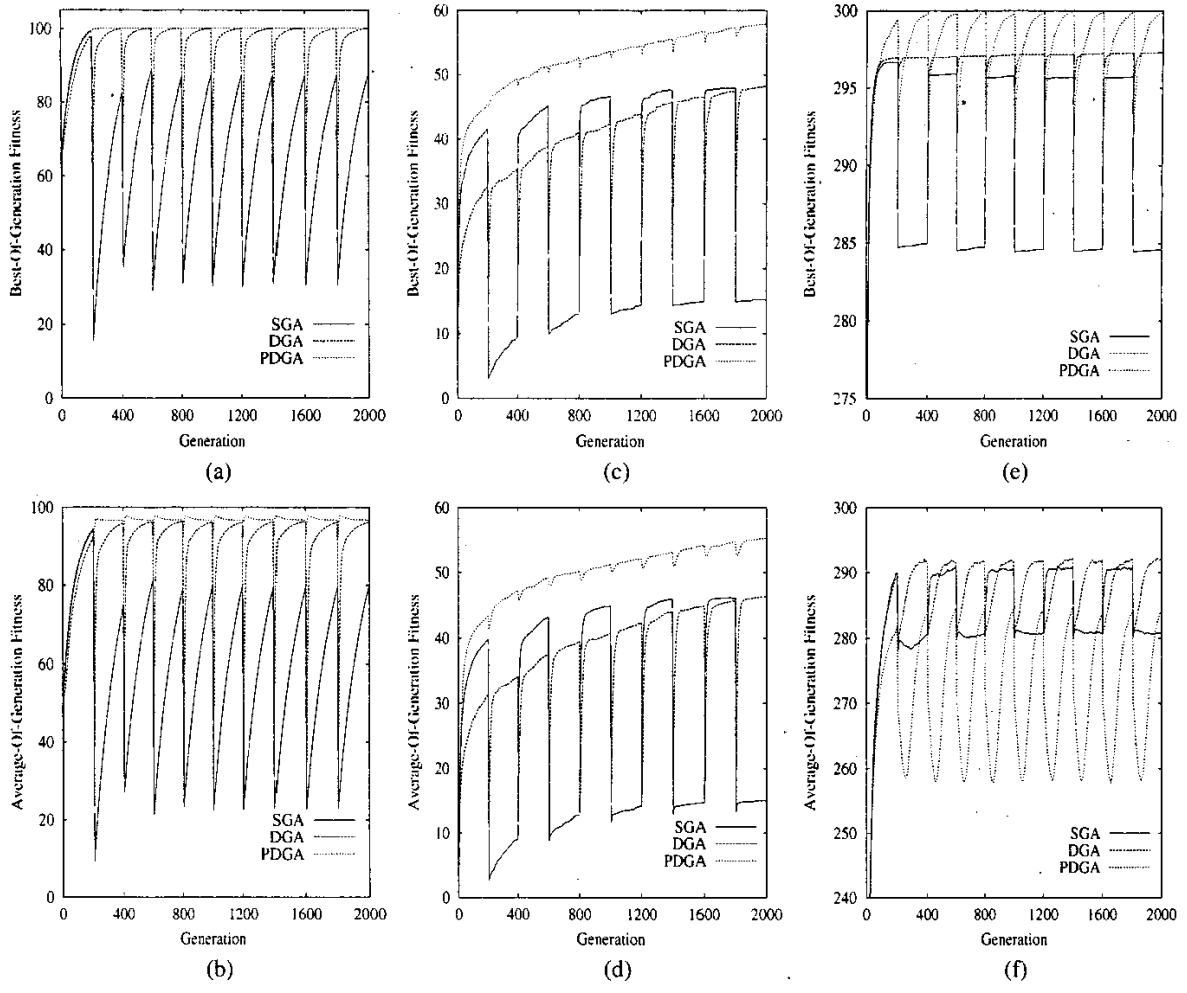

Figure 4: Experimental results with respect to best-of-generation fitness (*Top*) and average-of-generation fitness (*Bottom*) against generations of GAs on periodically reversing dynamic test problems: One-Max (*a* & *b*), Royal Road (*c* & *d*), and Deceptive Function (*e* & *f*).

it is less than that with SGA and DGA. The reason is similarly explained on the randomly shifting Deceptive Function. With SGA the phenomenon of performance oscillating happens again for the similar reason as explained above on the Royal Road function.

Both Figure 3 and Figure 4 show that PDGA has strong robustness and adaptability under dynamic environments, especially when the environment change is significant. This is due to the complementarity and dominance mechanisms devised in PDGA. The meta-bit scheme used in DGA also improves its adaptability under dynamic environments. However, the effect is limited due to the lacking of dominance mechanism or the blindness in applying the mechanism of complementarity.

5 Conclusions

Inspired by the complementarity and dominance mechanisms in nature, our proposed Primal-Dual Genetic Algorithm (PDGA) operates on a pair of chromosomes that are primal-dual to each other in the sense of maximum distance in a given distance space in genotype. During the running of PDGA before next generation starts, a set of low fit primal chromosomes are selected to evaluate their duals for expressing potential superior duals into next generation. This paper presents a new adaptive scheme of selecting primal chromosomes for evaluating their duals, which further improves PDGA's performace.

In this paper, we investigate an important aspect of PDGA, which lies in that it can adapt to dynamic environments efficiently. Using the dynamic problem generating technique proposed in this paper a set of dynamic problems is generated from a typical set of stationary problems as the algorithm test environments. Experimental study over these dynamic problems suggests that PDGA can solve complex dynamic problems more efficiently than traditional GA and Collard and his co-workers' Dual GA. The experimental results show that PDGA has strong viability and robustness

in dynamic environments, especially when the environment change is significant. We conclude that PDGA is a novel idea that can act as a dynamic problem optimizer.

The mechanism of prima-dual chromosomes in PDGA is quite general and hence can be generalized to other optimization methods, such as hill-climbing methods, to improve their capability in non-stationary environments, which is an interesting future work.

Bibliography

[Ackley87] D. H. Ackley (1987). *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Kluwer Academic Publishers.

[Baker87] J. E. Baker (1987). Reducing Bias and Inefficiency in the Selection Algorithms. In J. J. Grefenstelle (ed.), *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, 14-21. Lawrence Erlbaum Associates.

[CA94] P. Collard and J. P. Aurand (1994). DGA: An Efficient Genetic Algorithm. In A. Cohn (ed.), *Proc. of the 11th Europ. Conf. on Artificial Intelligence*, 487-491.

[CE96] P. Collard and C. Escazut (1996). Fitness Distance Correlation in a Dual Genetic Algorithm. In W. Wahlster (ed.), *Proc. of the 12th European Conf. on Artificial Intelligence*, 218-222.

[DeJong75] K. A. De Jong (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD Thesis, University of Michigan, Ann Abor.

[FM93] S. Forrest and M. Mitchell (1993). Relative Building-Block Fitness and the Building-Block Hypothesis. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2*, 109-126. Morgan Kaufmann Publishers.

[GS87] D. E. Goldberg and R. E. Smith (1987). Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy. In J. J. Grefenstelle (ed.), *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, 59-68. Lawrence Erlbaum Associates.

[Goldberg89a] D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

[Goldberg89b] D. E. Goldberg (1989). Genetic Algorithms and Walsh Functions: Part I, a Gentle Introduction. *Complex Systems*, **3**: 129-152.

[Holland75] J. H. Holland (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, University of Michigan Press.

[LHR98] J. Lewis, E. Hart and G. Ritchie (1998). A Comparison of Dominance Mechanisms and Simple Mutation on Non-Stationary Problems. In A. E. Eiben, T. Bäck, M. Schoenauer and H.-P. Schwefel (eds.), *Proc. of the 5th Int. Conf. on Parallel Problem Solving from Nature*, 139-148.

[MFH92] M. Mitchell, S. Forrest and J. H. Holland (1992). The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. In F. J. Varela and P. Bourgine (eds.), *Proc. of the 1st European Conference on Artificial Life*, 245-254. MIT Press.

[NW95] K. P. Ng and K. C. Wong (1995). A New diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimisation. In L. J. Eshelman (ed.), *Proc. of the 6th Int. Conf. on Genetic Algorithms*. Morgan Kaufmann Publishers.

[SW99] C. Stephens and H. Waelbroeck (1999). Schemata Evolution and Building Blocks. *Evolutionary Computation*, **7**(2): 109-128.

[Whitley91] L. D. Whitley (1991). Fundamental Principles of Deception in Genetic Search. In G. J. E. Rawlins (ed.), *Foundations of Genetic Algorithms 1*, 221-241.

[Yang03] S. Yang (2003). PDGA: the Primal-Dual Genetic Algorithm. In *Proc. of the 3rd Int Conf. on Hybrid Intelligent Systems*. IOS Press.