Dual-population Genetic Algorithm for Nonstationary Optimization

Taejin Park
Pusan National University
Dept. of Computer Engineering
Jangjeon-Dong San 30
Gumjeong-Gu, Busan
+82-51-510-3531
parktj@pusan.ac.kr

Ri Choe
Pusan National University
Dept. of Computer Engineering
Jangjeon-Dong San 30
Gumjeong-Gu, Busan
+82-51-510-3531
choilee@pusan.ac.kr

Kwang Ryel Ryu
Pusan National University
Dept. of Computer Engineering
Jangjeon-Dong San 30
Gumjeong-Gu, Busan
+82-51-510-2453
krryu@pusan.ac.kr

ABSTRACT
In order to solve nonstationary optimization problems efficiently, evolutionary algorithms need sufficient diversity to adapt to environmental changes. The dual-population genetic algorithm (DPGA) is a novel evolutionary algorithm that uses an extra population called the *reserve* population to provide additional diversity to the *main* population through crossbreeding. Preliminary experimental results on various periods and degrees of environmental change have shown that the distance between the two populations of DPGA is one of the most important factors that affect its performance. However, it is very difficult to determine the best population distance without prior knowledge about the given problem. This paper proposes a new DPGA that uses two reserve populations (DPGA2). The reserve populations are at different distances from the main population. The information inflow from the reserve populations is controlled by survival selection. Experimental results show that DPGA2 shows a better performance than other evolutionary algorithms for nonstationary optimization problems without relying on prior knowledge about the problem.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms
Algorithms

Keywords
Genetic algorithm, multi-population GA, dual-population GA, nonstationary optimization, dynamic optimization

1. INTRODUCTION
Since the population of a genetic algorithm (GA) offers the advantage of diversity, the GA adapts more easily to environmental changes, and consequently, shows a better performance for dynamic optimization as compared to other meta-heuristic algorithms such as hill-climbing, simulated annealing, and tabu search. However, as the population evolves, the GA loses its diversity and it can no longer adapt easily to environmental changes. Many previous studies have proposed various methods to cope with this problem. Branke [2] has grouped these methods into four categories: increasing diversity after a change [3, 11], maintaining diversity throughout the run [4], memory-based methods [5, 7] and multi-population approaches [1, 10].

The dual-population genetic algorithm (DPGA) was originally proposed for stationary optimization problems [8, 9]. It employs two distinct populations with different evolutionary objectives. The *main population* plays the same role as that of the population of an ordinary GA. It evolves to find a good solution with a high fitness value. The additional population called the *reserve population* provides additional diversity to the main population. In order to allow the main population to use the diversity in the reserve population, there must be a method of exchanging information between the populations. The migration method used for most multi-population GAs (MPGAs), however, is not suitable for DPGA because the two populations have different evolutionary objectives. An individual of one population can hardly survive in the other because the methods for evaluating fitness are different for both populations. Therefore, DPGA uses *crossbreeding* as a means of information exchange. In DPGA, offspring are produced by mating not only the individuals of the same population but also the individuals of different populations. Since the crossbred offspring contain the genetic material of both populations, their fitness values are not too low and thus they assimilate relatively easily into the new population.

The population distance is a very important factor that affects the performance of DPGA. The distance between the populations of DPGA can be controlled by the fitness function parameter of the reserve population [9]. The best distance depends on the characteristics of a given problem. For example, DPGA must maintain a rather small distance between the main population and the reserve populations for most stationary optimization problems. However, for problems that require high diversity, such as the deceptive problems, the algorithm must maintain a relatively large distance between the populations. For dynamic optimization problems, the best distance between the populations depends on the dynamic characteristics of the given problems, such as the period of change and the degree of change. In this study, we apply DPGA to dynamic optimization problems of various periods and degrees of change in order to identify the relationship between the best population distance and the dynamic characteristics. In addition, we propose a new DPGA with two reserve populations for dynamic optimization problems. The two reserve populations are at different distances from the main population and the information inflow from the reserve populations is controlled by survival selection.

The rest of the paper is organized as follows. Section 2 explains the DPGA in detail and section 3 describes the new version of DPGA that has two reserve populations. Section 4 reports the experimental results with various dynamic optimization problems and compares DPGA with other genetic algorithms. Finally, section 5 provides conclusions.

2. Dual-population Genetic Algorithm

2.1 Fitness Function for Reserve Population

DPGA manipulates two populations with different evolutionary objectives. The individuals of each population are evaluated by the fitness function of the population based on its evolutionary objective. The fitness function $fm(\mathbf{x})$ of the main population is the same as the evaluation function given by the problem, as its evolutionary objective is to find a good solution. The individuals of the reserve population are evaluated by their average distance to the individuals of the main population, because the evolutionary objective of the reserve population is to provide additional diversity to the main population. Previous studies [9] have used equation (1) as the fitness function of the reserve population so that an individual has its maximum fitness value of 1 when its average distance to the individuals of the main population is equal to a given value δ:

$$fr_{\delta}(\mathbf{x}) = 1 - |\delta - d(\mathbf{M}, \mathbf{x})| \tag{1}$$

where $0 \le \delta \le 1$, and $0 \le d(\mathbf{M}, \mathbf{x}) \le 1$ is the normalized average distance between the individuals of the main population $\mathbf{M}$ and an individual $\mathbf{x}$ of the reserve population. Assuming a binary representation for a chromosome, $d(\mathbf{M}, \mathbf{x})$ is calculated using equation (2):

$$d(\mathbf{M}, \mathbf{x}) = \frac{1}{|\mathbf{M}|} \sum_{\mathbf{m} \in \mathbf{M}} dist(\mathbf{m}, \mathbf{x}) = \frac{1}{l} \sum_{k=1}^{l} |f_{\mathbf{M},k} - x_k| \tag{2}$$

where $|\mathbf{M}|$ is the size of the main population $\mathbf{M}$, l is the length of a chromosome, and $dist(\mathbf{m}, \mathbf{x})$ is the distance between two chromosome vectors $\mathbf{m}$ and $\mathbf{x}$. $dist(\mathbf{m}, \mathbf{x})$ can be the Hamming distance in case of binary representation. $f_{\mathbf{M},k}$ represents the frequency of the kth gene value "1" of $\mathbf{M}$. x_k denotes the frequency of the kth gene value "1" of the chromosome vector $\mathbf{x}$ and is identical to the kth gene value of the chromosome. The details of this function can be found in previous studies [8, 9]. When $\delta = 0$, individuals that are very similar to those of the main population have high fitness values. In contrast, when $\delta = 1$, individuals that are very different from those of the main population have high fitness values.

2.2 Algorithm for Evolution

DPGA begins with two randomly generated populations. The individuals of each population are evaluated by the fitness function of the population to which they belong. Ordinary MPGAs generate new offspring only by *inbreeding*, i.e., a recombination between parents selected from the same population. However, DPGA generates offspring by both inbreeding and crossbreeding. The *inbred offspring* and *crossbred offspring* compete with each other through *survival selection* and only the winners are selected for the next generation. Figure 1 shows the pseudocode of DPGA.

DPGA generates n offspring for each population and then the best m offspring are selected for the next generation of each population ($n > m$). First, the algorithm generates m inbred offspring $\mathbf{I}_M$ through inbreeding between parents from the main population. Similarly, the algorithm generates m inbred offspring $\mathbf{I}_R$ through inbreeding between parents from the reserve population. Next, the algorithm generates $(n - m)$ crossbred offspring $\mathbf{C}$ through crossbreeding. The inbred offspring $\mathbf{I}_M$ of the main population and the crossbred offspring $\mathbf{C}$ constitute a candidate set $\mathbf{O}_M$ for the next generation of the main population. Similarly, the inbred offspring $\mathbf{I}_R$ of the reserve population and the crossbred offspring $\mathbf{C}$ constitute a candidate set $\mathbf{O}_R$ for the reserve population.

The procedure *GenerateInbredOffspring* for generating inbred offspring is identical to the procedure of a standard GA for generating offspring. Two parents are selected from the given population $\mathbf{M}^t$ or $\mathbf{R}^t$ according to their fitness values and new offspring are generated by using crossover and mutation. The procedure *GenerateCrossbredOffspring* for generating crossbred offspring selects the parents from different populations, one from the main population and the other from the reserve population. It should be noted that the fitness functions of the two populations are different; hence, the basis for selection of the parents is different. After selecting the parents, new offspring are generated by crossover and mutation. Since the crossbred offspring contain genetic materials of both the parents, they can function as a medium of information exchange.

The next step is survival selection for the next generation of each population. First, the individuals in the candidate set $\mathbf{O}_M$ are evaluated by the fitness function $fr(\mathbf{x})$ of the main population. Then, the algorithm selects m individuals from $\mathbf{O}_M$ for the next genera-

```
Procedure DPGA
begin
    Initialize population $\mathbf{M}^0$ and $\mathbf{R}^0$  ($|\mathbf{M}^0| = |\mathbf{R}^0| = m$)
    Evaluate $\mathbf{M}^0$ using $fm(\mathbf{x})$
    Evaluate $\mathbf{R}^0$ using $fr(\mathbf{x})$
    $t := 0$
    repeat
        $\mathbf{I}_M$ = GenerateInbredOffspring($\mathbf{M}^t$, $m$)
        $\mathbf{I}_R$ = GenerateInbredOffspring($\mathbf{R}^t$, $m$)
        $\mathbf{C}$ = GenerateCrossbredOffspring($\mathbf{M}^t$, $\mathbf{R}^t$, $n - m$)
        $\mathbf{O}_M = \mathbf{I}_M \cup \mathbf{C}$
        $\mathbf{O}_R = \mathbf{I}_R \cup \mathbf{C}$
        Evaluate $\mathbf{O}_M$ using $fm(\mathbf{x})$
        Evaluate $\mathbf{O}_R$ using $fr_\delta(\mathbf{x})$
        $\mathbf{M}^{t+1}$ = SurvivalSelection($\mathbf{O}_M$, $m$)
        $\mathbf{R}^{t+1}$ = SurvivalSelection($\mathbf{O}_R$, $m$)
        $t := t + 1$
    until terminated = true
end
```

Figure 1. Pseudocode of DPGA.

tion $\mathbf{M}^{t+1}$ of the main population through survival selection. Similarly, the individuals in the candidate set $\mathbf{O}_R$ are evaluated by the fitness function $fr_\delta(\mathbf{x})$, and m individuals are selected from $\mathbf{O}_R$ for the next generation $\mathbf{R}^{t+1}$ of the reserve population. For survival selection, individuals in the given candidate set are sorted by their fitness values and the best m individuals are selected. Since m is equal to the number of generated inbred offspring, the crossbred offspring can survive only if they are better than at least one of the inbred offspring. If too much genetic materials are imported through crossbred offspring, the convergence of the population can be disturbed and it could be difficult for the population to evolve into a good solution. Therefore the influx of the crossbred offspring is controlled by survival selection.

3. DPGA with two reserve populations

The fitness function $fr_\delta(\mathbf{x})$ drives the individuals of the reserve population to evolve to ones whose distance to the main population is δ. When δ is too small, the individuals of the reserve population become very similar to those of the main population and therefore do not provide sufficient diversity. On the contrary, when δ is too large, the parents for crossbreeding are too different from each other and therefore do not give birth to good offspring. When the parents for crossbreeding are genetically very different from each other, the probability that the offspring will be much worse than their parents is high. Accordingly, as the populations converge and their average population fitness attains high values, it becomes difficult for the crossbred offspring to win the survival selection. In such cases, the reserve population also cannot provide sufficient diversity to the main population because genetic materials are rarely exchanged between the populations.

A previous study [9] shows that the performance of DPGA is highly dependent on the value of δ, and the best value of δ differs for each problem. In general, DPGA shows a good performance for stationary problems when the value of δ is rather small (e.g., 0.1). Exceptionally, a large value of δ is required for the deceptive problems because they demand high diversity to be solved efficiently. Due to the difficulty in determining the best value of δ, the previous study proposed a method for adjusting the value of δ by using the information obtained through evolution.

For nonstationary optimization problems, the best value of δ is affected by the dynamic characteristics of a problem such as the speed of change and degree of change. According to our preliminary experiments, DPGA shows a good performance for a dynamic optimization problem with a small degree of change when the value of δ is small. However, a larger value of δ is required for a problem with a higher degree of change. The method for adjusting the value of δ proposed in the previous study is not suitable for nonstationary optimization problems. Since the method is devised for stationary problems, it tends to maintain a low average value of δ even when it is applied to nonstationary problems. Consequently, the reserve population cannot maintain sufficient diversity for a problem with a high degree of change.

In this paper, we propose a new algorithm called DPGA2 in which the reserve population is split into two independent subpopulations $\mathbf{R}_1$ and $\mathbf{R}_2$, with $\mathbf{R}_1$ having a small value δ_1 and $\mathbf{R}_2$ a large value δ_2. Crossbred offspring are generated only between the main population and each of the reserve populations. Since it is not necessary to exchange information between the reserve populations, no crossbred offspring are generated between them. Each crossbred offspring competes with both inbred offspring and other crossbred offspring, and thus, the inflow of genetic material from each reserve population is automatically adjusted by survival selection. We expect a good amount of information flow into the main population from $\mathbf{R}_1$ for problems with a low degree of change or from $\mathbf{R}_2$ for problems with a high degree of change.

The algorithm of DPGA2 is shown in Figure 2. DPGA2 generates n offspring and selects the best m offspring for the main population. However, it generates $n/2$ offspring and selects the best $m/2$ offspring for each reserve population. The main population $\mathbf{M}$ generates m inbred offspring $\mathbf{I}_M$, and the reserve population R_1 and R_2 generate $m/2$ inbred offspring $\mathbf{I}_{R1}$ and $\mathbf{I}_{R2}$, respectively. Then, $(n-m)/2$ offspring $\mathbf{C}_1$ are generated by crossbreeding between $\mathbf{M}$ and $\mathbf{R}_1$. Similarly $(n-m)/2$ offspring $\mathbf{C}_2$ are generated by crossbreeding between $\mathbf{M}$ and $\mathbf{R}_2$. The inbred offspring $\mathbf{I}_M$ of the main population and both the crossbred offspring $\mathbf{C}_1$ and $\mathbf{C}_2$ become candidates $\mathbf{O}_M$ for the next generation of the main population. The inbred offspring $\mathbf{I}_{R1}$ and the crossbred offspring $\mathbf{C}_1$ become candidate $\mathbf{O}_{R1}$ for the reserve population $\mathbf{R}_1$, and the inbred offspring $\mathbf{I}_{R2}$ and the crossbred offspring $\mathbf{C}_2$ become candidates $\mathbf{O}_{R2}$ for the reserve population $\mathbf{R}_2$. Finally, the candidate set of each population is evaluated by its fitness function and the best m or $m/2$ individuals are selected for the next generation through survival selection.

```
Procedure DPGA2
begin
  Initialize population $\mathbf{M}_0$, $\mathbf{R}_1^0$, and $\mathbf{R}_2^0$
        $(|\mathbf{M}^0| = m, |\mathbf{R}_1^0| = |\mathbf{R}_2^0| = m/2)$
  Evaluate $\mathbf{M}^0$ using $fm()$
  Evaluate $\mathbf{R}_1^0$ using $fr_{\delta 1}()$
  Evaluate $\mathbf{R}_2^0$ using $fr_{\delta 2}()$
  $t := 0$
  repeat
    $\mathbf{I}_M :=$ GenerateInbredOffspring($\mathbf{M}^t, m$)
    $\mathbf{I}_{R1} :=$ GenerateInbredOffspring($\mathbf{R}_1^t, m/2$)
    $\mathbf{I}_{R2} :=$ GenerateInbredOffspring($\mathbf{R}_2^t, m/2$)
    $\mathbf{C}_1 :=$ GenerateCrossbredOffspring($\mathbf{M}^t, \mathbf{R}_1^t, (n-m)/2$)
    $\mathbf{C}_2 :=$ GenerateCrossbredOffspring($\mathbf{M}^t, \mathbf{R}_2^t, (n-m)/2$)
    $\mathbf{O}_M := \mathbf{I}_M \cup \mathbf{C}_1 \cup \mathbf{C}_2$
    $\mathbf{O}_{R1} := \mathbf{I}_{R1} \cup \mathbf{C}_1$
    $\mathbf{O}_{R2} := \mathbf{I}_{R2} \cup \mathbf{C}_2$
    Evaluate $\mathbf{O}_M$ using $fm(\mathbf{x})$;
    Evaluate $\mathbf{O}_{R1}$ using $fr_{\delta 1}(\mathbf{x})$
    Evaluate $\mathbf{O}_{R2}$ using $fr_{\delta 2}(\mathbf{x})$
    $\mathbf{M}^{t+1} :=$ SurvivalSelection($\mathbf{O}_M, m$)
    $\mathbf{R}_1^{t+1} :=$ SurvivalSelection($\mathbf{O}_{R1}, m/2$)
    $\mathbf{R}_2^{t+1} :=$ SurvivalSelection($\mathbf{O}_{R2}, m/2$)
    $t := t + 1$
  until terminated = true
end
```

Figure 2. Pseudocode of DPGA with two reserve populations (DPGA2).

4. Experimental results

We have used the dynamic problem generator that was proposed by [14, 15]. This generator can create various dynamic problems using two parameters: the period of change τ and the degree of change ρ. Given a binary-encoded stationary optimization problem $f(\mathbf{x})$, the dynamic problem $f(\mathbf{x}, t)$ is defined as follows.

$$f(\mathbf{x}, t) = f(\mathbf{x} \oplus \mathbf{m}(k)) \tag{3}$$

where t is the generation count, $k = \lfloor t/\tau \rfloor$ is the period index, τ is the length of the period, and $\mathbf{m}(k)$ is a binary mask for the kth period. In order to evaluate an individual $\mathbf{x}$, we first perform the operation $\mathbf{x} \oplus \mathbf{m}(k)$, where "$\oplus$" is the bitwise exclusive-or (XOR) operator, and then evaluate the resulting individual. The first mask $\mathbf{m}(0)$ is randomly generated, and the mask $\mathbf{m}(k)$ of the kth period is generated incrementally using the mask $\mathbf{m}(k-1)$ of the previous period, as shown below.

$$\mathbf{m}(k) = \mathbf{m}(k-1) \oplus \mathbf{t}(k) \tag{4}$$

where $\mathbf{t}(k)$ is an intermediate binary template randomly generated for the period k containing $\rho \times l$ ones. When $\rho = 0.0$, the problem stays stationary, and a larger ρ causes a severer change. Also, $\mathbf{m}(k)$ changes more frequently as τ gets smaller.

This paper uses three well-known problems as the base problem $f(\mathbf{x})$: binary knapsack problem, royal road function [6], and deceptive function [12]. A knapsack problem with 50 items is randomly generated using the method described in [15]. To see the effects of the dynamic parameters, the period of change τ is set to 10, 100, and 200 generations, and the degree of change ρ is set to 0.05, 0.25, 0.5, 0.75, and 0.95. In total, a series of 15 dynamic problems is constructed from each base problem.

First, we compare DPGA2 with DPGA using various values of δ. For DPGA2, we set δ_1 to 0.1 and δ_2 to 0.9. For DPGA, δ is set to 0.1, 0.5, and 0.9. The parent size m is set to 80 and the offspring size n is set to 100. We use a two-point crossover with a crossover rate of 1.0 and bitwise mutation with a mutation rate of $1/l$, where l is the length of a chromosome. Tournament selection together with elitism is used for selection.

For each experiment, 100 independent runs are executed and each algorithm is run for 1,000 generations. For each run of the various algorithms on each problem, the best-of-generation fitness is recorded for every generation. Figure 3 shows the best-of-generation fitness of the algorithms on the stationary problems. The best value of δ is 0.1 for the knapsack problem and 0.9 for the royal road and deceptive functions. The best value of δ differs for each problem; however, DPGA2 shows curves similar to those of DPGA with the best values of δ for all the stationary problems.

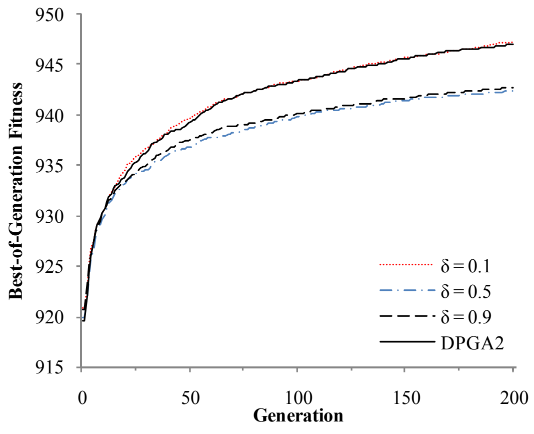
For the dynamic problems, the overall performance is measured by the mean best-of-generation fitness $\overline{BGF}$. It is defined as the best-of-generation fitness averaged across all the different runs and then averaged over all the generations.

$$\overline{BGF} = \frac{1}{G} \cdot \sum_{i=1}^{G} \left(\frac{1}{N} \cdot \sum_{j=1}^{N} BGF_{ij} \right) \tag{5}$$

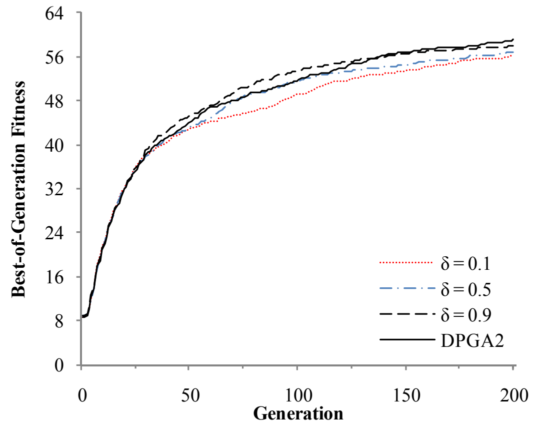
where G is the number of generations, N is the total number of runs, and BGF_{ij} is the best-of-generation fitness of generation i of run j.

Table 1. The mean best-of-generation fitness of the DPGA with δ = 0.1, 0.5, and 0.9 and DPGA2 on the dynamic problems.

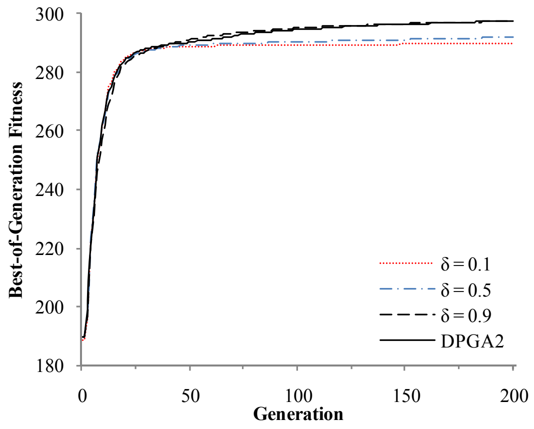
Param. Setting Index	Parameters Setting (τ, ρ)	Knapsack Problem				Royal Road function				Deceptive function			
		$\delta = 0.1$	$\delta = 0.5$	$\delta = 0.9$	DPGA2	$\delta = 0.1$	$\delta = 0.5$	$\delta = 0.9$	DPGA2	$\delta = 0.1$	$\delta = 0.5$	$\delta = 0.9$	DPGA2
1	(10, 0.05)	**933.4**	930.8	930.5	934.0	**39.0**	37.8	37.3	38.9	**279.0**	276.3	275.3	277.7
2	(10, 0.25)	926.2	**927.6**	927.2	927.7	12.3	**16.1**	15.8	15.4	236.6	**238.7**	238.0	237.9
3	(10, 0.50)	922.3	925.6	**925.7**	925.3	8.8	12.7	**12.8**	11.9	220.2	**225.6**	225.2	223.8
4	(10, 0.75)	918.5	923.7	**924.4**	923.4	9.4	12.7	**13.1**	12.2	232.4	234.6	**234.5**	234.0
5	(10, 0.95)	913.3	921.3	**922.6**	920.3	**18.5**	16.8	17.3	18.3	**278.3**	273.8	269.8	275.3
6	(100, 0.05)	**946.2**	938.5	938.5	944.5	**57.1**	56.7	56.4	57.4	290.3	292.8	**293.2**	295.5
7	(100, 0.25)	**939.7**	936.8	936.6	939.8	41.9	**43.4**	43.0	43.1	283.1	**284.5**	286.2	286.2
8	(100, 0.50)	934.9	**936.0**	935.8	937.2	26.8	37.4	**37.5**	35.9	280.1	281.6	**283.2**	282.0
9	(100, 0.75)	931.6	935.6	**936.3**	937.2	23.7	35.8	**41.7**	39.8	283.4	283.8	**285.5**	284.5
10	(100, 0.95)	929.7	935.3	**937.5**	939.7	28.0	35.1	**53.0**	51.8	288.7	288.4	**291.0**	290.3
11	(200, 0.05)	**947.6**	940.7	940.4	946.4	57.7	**58.2**	57.7	58.2	290.9	293.9	**295.2**	296.8
12	(200, 0.25)	**943.4**	939.2	939.1	943.0	49.6	**49.9**	49.7	50.0	287.6	289.0	**291.5**	291.7
13	(200, 0.50)	**939.5**	938.8	938.7	941.3	36.2	45.8	**46.0**	44.7	285.1	286.3	**290.4**	289.2
14	(200, 0.75)	936.6	938.3	**938.9**	941.6	29.1	45.0	**49.1**	47.5	286.8	287.2	**291.4**	290.1
15	(200, 0.95)	934.7	938.4	**939.9**	943.3	32.4	44.6	**56.4**	56.0	289.0	289.1	**294.2**	293.4



(a)

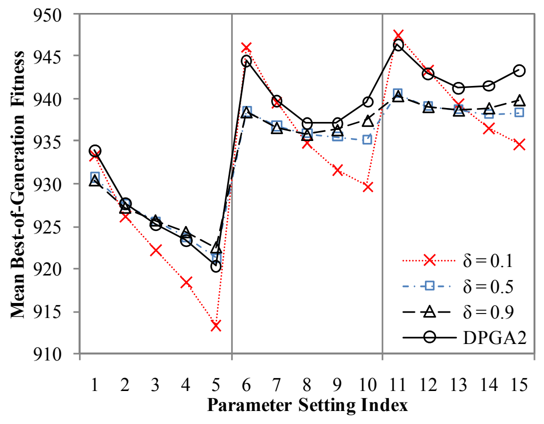


(b)

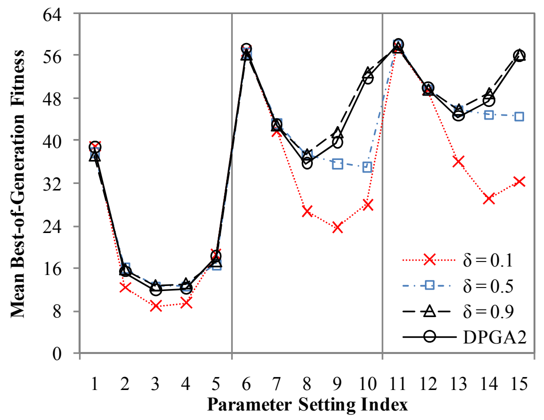


(c)

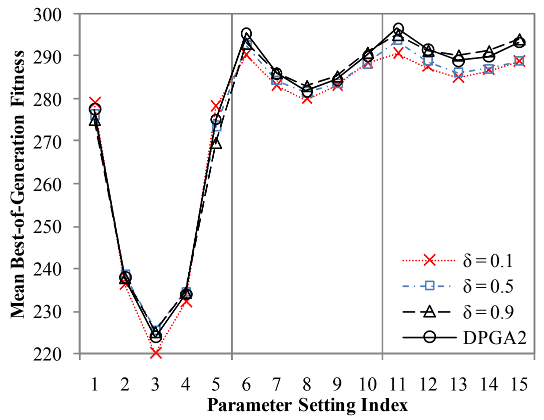
Figure 3. The best-of-generation fitness of the algorithms on the stationary problems: (a) knapsack, (b) royal road, and (c) deceptive problem.



(a)



(b)



(c)

Figure 4. The mean best-of-generation fitness of the DPGA with δ = 0.1, 0.5, and 0.9 and DPGA2 on the dynamic problems: (a) knapsack, (b) royal road, and (c) deceptive problem.

Table 1 and Figure 4 show the results obtained for the dynamic problems. For the dynamic knapsack problems, DPGA with a small value of δ shows the best performance when the degree of change (ρ) is small. As ρ gets larger, however, the algorithm shows the best performance with a larger value of δ. When the period of change (τ) is 10, we can see from Table 1 that the best value of δ for DPGA is 0.1 for ρ = 0.05, but the best δ becomes 0.9 for ρ = 0.95. In fact, similar observations can be made with the other values of τ. A more careful examination of the data reveals that there is a tendency that DPGA shows its best performance at a smaller value of δ as the period of change (τ) gets longer. For example, when ρ = 0.5, the best values of δ are 0.9, 0.5, and

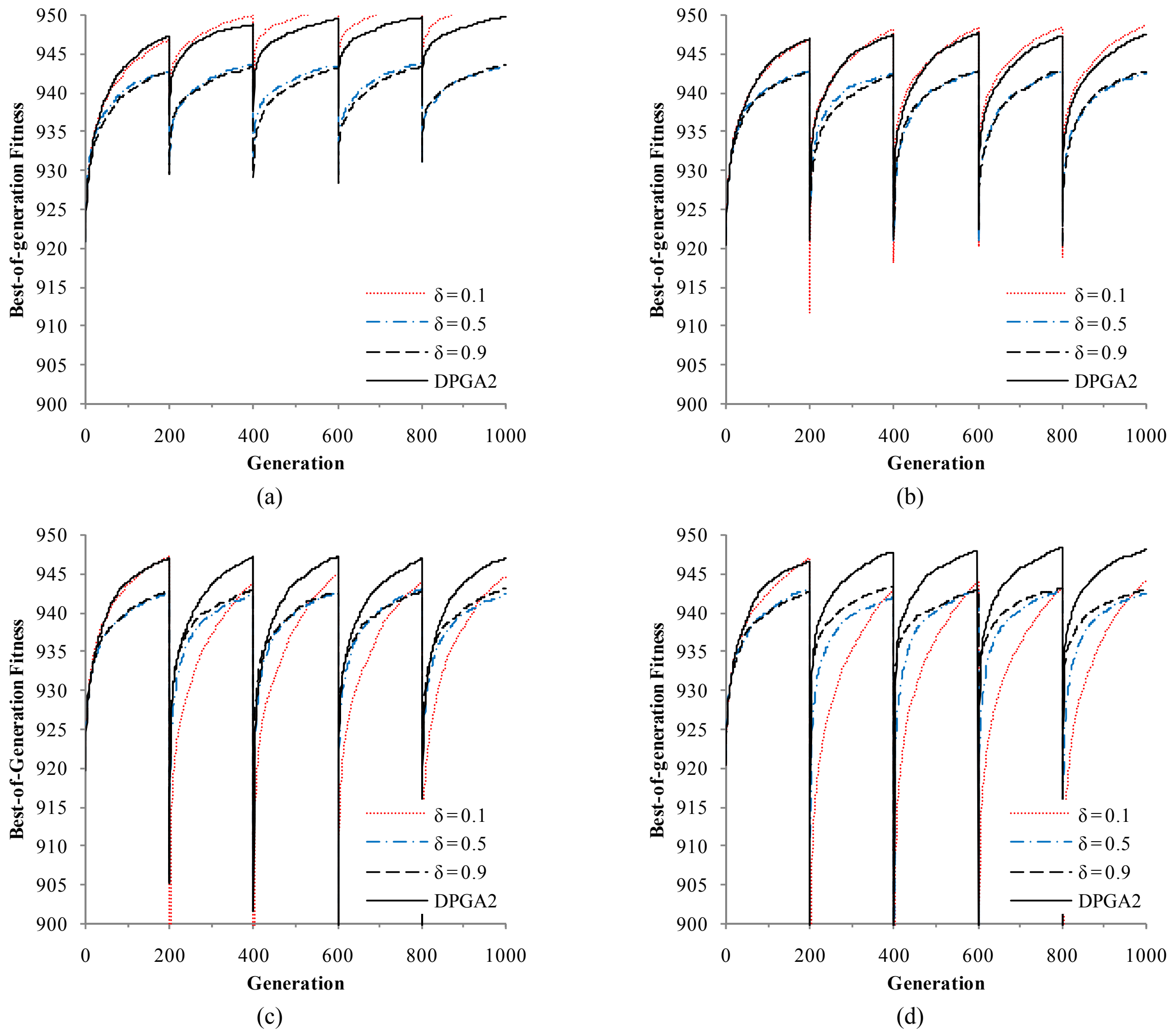


Figure 5. Dynamic behavior of the DPGA with various δ and DPGA2 on dynamic knapsack problems when $\tau = 200$. ρ is set to (a) 0.05, (b) 0.25, (c) 0.75, and (d) 0.95.

0.1 for τ = 10, 100, and 200, respectively. This seems to imply that DPGA demands more diversity as the period of change τ gets shorter. Note that DPGA2 shows performances close to those of DPGA using its best values of δ for every setting of dynamic parameters τ and ρ. For some cases, e.g., (200, 0.75) and (200, 0.95), the performance of DPGA2 is even significantly better than that of DPGA with its best δ.

For the dynamic royal road functions, all the algorithms show similar performances when ρ is small. When ρ is large, $\delta = 0.9$ is the best for DPGA in most cases, and DPGA2 gives results close to those of DPGA with $\delta = 0.9$.

In case of the dynamic deceptive problems, DPGA with $\delta = 0.9$ shows the best performance for most dynamic settings, except when τ is very short. Note that for the deceptive problems, even when they are stationary, DPGA requires large diversity and thus its performance with $\delta = 0.9$ is much better than those with $\delta = 0.1$ and 0.5. The best value of δ for a dynamic problem seems to be highly dependent on the characteristics of its stationary version problem as well as on its dynamic parameter setting. Therefore, it is very difficult to determine the best value of δ without prior knowledge about the given problem or careful experiments. However, DPGA2 shows good performances close to those of DPGA with its best value of δ for almost all the problems and dynamic parameter settings.

Figure 5 shows the dynamic behavior of DPGA with various δ values and that of DPGA2 on dynamic knapsack problems for the dynamic parameter settings of τ = 200 and ρ = 0.05, 0.25, 0.75, and 0.95. When ρ is 0.05 and 0.25, DPGA with $\delta = 0.1$ shows the best performance by converging fast during the first period and adapting easily to environmental changes from then on through the following periods. However, when ρ = 0.75 and 0.95, DPGA with $\delta = 0.1$ shows the worst performance. Although it converges fast during the first period, it does not adapt easily to severe environmental changes. Perhaps its reserve population may not maintain sufficient diversity to adapt to such extreme environmental changes. As a result, DPGA with $\delta = 0.1$ shows much worse performance from the second period. On the other hand, although DPGA with $\delta = 0.9$ converges slowly, it shows more robust performance against environmental changes. We can see that the

Table 2. The mean best-of-generation fitness of SGA, PDGA, IMGA, and DPGA2 on dynamic problems.

Param. Setting Index	Parameters Setting (τ, ρ)	Knapsack problem				Royal Road function				Deceptive problem			
		SGA	PDGA	IMGA	DPGA2	SGA	PDGA	IMGA	DPGA2	SGA	PDGA	IMGA	DPGA2
1	(10, 0.05)	933.1	933.0	**934.4**	934.0	37.4	37.7	36.7	**38.9**	275.4	275.6	276.4	**277.7**
2	(10, 0.25)	927.3	927.4	**927.7**	**927.7**	13.7	13.9	13.1	**15.4**	234.3	234.4	233.9	**237.9**
3	(10, 0.50)	924.3	924.4	924.2	**925.3**	10.3	10.4	10.0	**11.9**	218.9	218.8	218.0	**223.8**
4	(10, 0.75)	921.7	921.8	921.1	**923.4**	10.7	11.0	10.4	**12.2**	229.8	229.7	229.0	**234.0**
5	(10, 0.95)	917.7	918.0	916.4	**920.3**	17.7	**18.3**	17.3	**18.3**	273.5	273.3	274.3	**275.3**
6	(100, 0.05)	941.6	941.4	943.8	**944.5**	56.8	56.5	55.6	**57.4**	291.3	292.9	292.1	**295.5**
7	(100, 0.25)	938.7	938.7	**939.9**	939.8	42.6	42.5	40.9	**43.1**	283.2	284.2	283.9	**286.2**
8	(100, 0.50)	936.2	936.3	937.0	**937.2**	31.8	32.0	29.8	**35.9**	279.6	280.1	279.5	**282.0**
9	(100, 0.75)	934.3	934.3	934.5	**937.2**	26.6	31.9	25.0	**39.8**	282.7	283.0	282.9	**284.5**
10	(100, 0.95)	932.7	933.0	932.4	**939.7**	25.7	49.1	25.4	**51.8**	288.2	288.7	288.4	**290.3**
11	(200, 0.05)	943.0	943.1	945.3	**946.4**	57.9	57.9	56.6	**58.2**	292.2	294.5	292.6	**296.8**
12	(200, 0.25)	941.2	941.3	**943.1**	943.0	**50.1**	49.9	47.3	50.0	287.7	289.5	288.5	**291.7**
13	(200, 0.50)	939.6	939.7	941.0	**941.3**	41.8	42.2	39.6	**44.7**	285.2	286.9	285.4	**289.2**
14	(200, 0.75)	938.7	938.7	939.4	**941.6**	36.7	43.1	34.2	**47.5**	286.3	288.3	286.5	**290.1**
15	(200, 0.95)	937.8	938.2	938.5	**943.3**	33.8	54.1	31.5	**56.0**	288.7	291.1	289.2	**293.4**

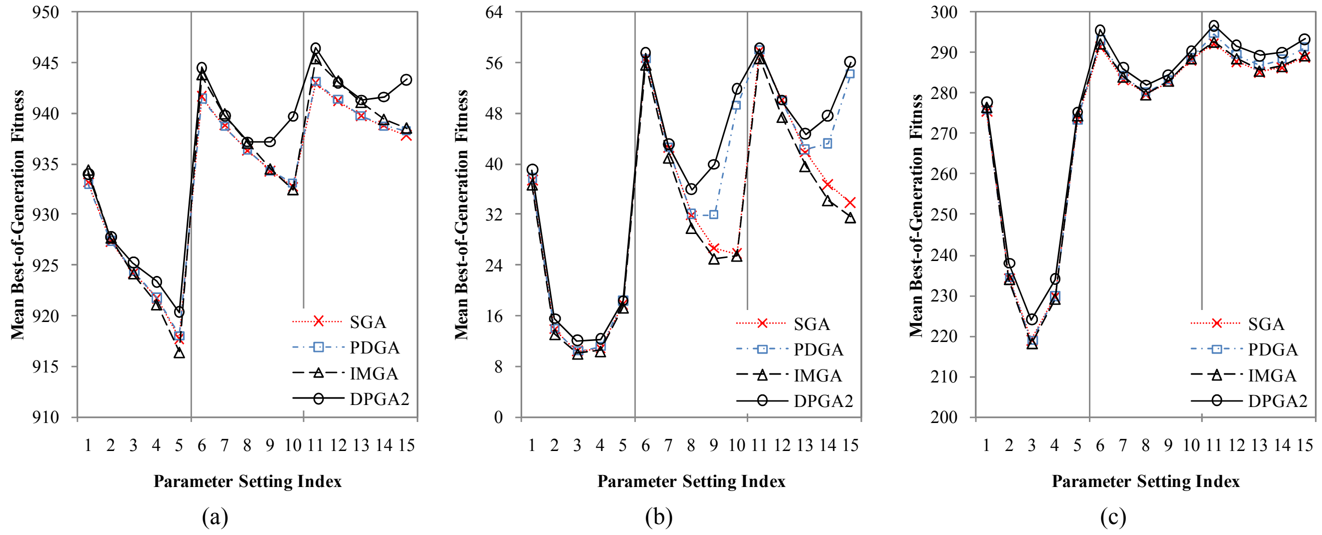


Figure 6. The mean best-of-generation fitness of SGA, PDGA, IMGA, and DPGA2 with different combinations of dynamic parameters on dynamic problems: (a) knapsack, (b) royal road, and (c) deceptive problem.

performance with $\delta = 0.9$ is worse than that with $\delta = 0.1$ when $\rho = 0.05$ and 0.25, but it converges steadily all through the periods, even when $\rho = 0.75$ and 0.95. DPGA2 shows fast convergence and adaptability to a small degree of change, as does DPGA with $\delta = 0.1$. DPGA2 also shows robustness to a high degree of environmental changes, as does DPGA with $\delta = 0.9$. Having the strengths of both DPGAs, DPGA2 performs better than all the DPGAs when $\rho = 0.75$ and 0.95.

Finally, we compare DPGA2 with a standard GA (SGA), a primal-dual GA (PDGA), and an island-model GA (IMGA). SGA is a general generative-model GA. PDGA adopts a complementary and dominance mechanism and uses a complementary chromosome—dual chromosome—to provide additional diversity to the population. Further details on PDGA can be found in [14]. IMGA is a typical multi-population GA that employs two distinct populations, evolves them separately, and exchanges some of their individuals regularly [13]. In our experiment, the best solution of each population is exchanged in every five generations. The population size is set to 100 for SGA and PDGA and to 50 + 50 for IMGA. Other genetic operators and parameters are set identical to those of DPGA and DPGA2.

Table 2 and Figure 6 compare the experimental results obtained by running these genetic algorithms. For most problems and most dynamic parameter settings, DPGA2 shows the best performance.

For knapsack problems, the curves of the other three algorithms are similar to each other, and their performances degrade as the degree of change increases. When $\tau = 10$, the curve of DPGA2 is not very different from the curves of the other algorithms. When $\tau = 100$ and 200, however, the curve of DPGA2 shows a different "U" shape due to the performance upgrades after $\rho = 0.5$. On dynamic royal road functions, PDGA also displays a similar "U" curve although not as good as that of DPGA2. On dynamic deceptive problems, all the algorithms show quite similar performance curves, although DPGA2 again shows the best performance for all the dynamic parameter settings.

5. Conclusions

DPGA is a novel evolutionary algorithm that uses a reserve population to provide additional diversity to the main population. The distance between the two populations is a very important factor that affects the performance of the algorithm and can be controlled by using the parameter δ of the fitness function for the reserve population. In this study, we extended DPGA to solve dynamic optimization problems. We first investigated the relationship between the value of δ and the dynamic characteristics such as the period and degree of change. Our experiments revealed that DPGA with a relatively small value of δ shows the best performance for a low degree of change by converging fast and adapting well to moderate environmental changes. However, DPGA with a larger value of δ shows a better performance for a high degree of change by providing enough diversity to cope with extreme environmental changes. The problem is that the best value of δ differs for each problem and each dynamic environmental characteristic; hence, it is difficult to decide the best value of δ without prior knowledge or intensive experiments on the given problem.

In this paper, we proposed a new algorithm called DPGA2 having its reserve population split into two. One of the reserve populations uses a small δ for the fitness function and the other uses a large δ. Since the inflow of genetic material from each reserve population to the main population is automatically adjusted by crossbreeding and survival selection, an appropriate amount of diversity can always be provided regardless of the problem characteristics. Experiments showed that the performance of DPGA2 is close to that of DPGA with small δ for a low degree of change and DPGA with large δ for a high degree of change. Additional experiments showed that DPGA2 is better than other evolutionary algorithms based on similar concepts.

6. ACKNOWLEDGMENTS

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (The Regional Research Universities Program/Research Center for Logistics Information Technology).

7. REFERENCES

[1] J. Branke, T. Kaubler, C. Schmidt, and H. Schmeck. A multi-population approach to dynamic optimization problems. In *Adaptive Computing in Design and Manufacturing 2000*. Springer, 2000.

[2] J. Branke. Evolutionary approaches to dynamic optimization problems—updated survey. *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 134–137, 2001.

[3] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. *Technical Report AIC-90-001*, Naval Research Laboratory, Washington, USA, 1990.

[4] J. J. Grefenstette. Genetic algorithms for changing environments. In R. Maenner and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 137–144, 1992.

[5] B. S. Hadad and C. F. Eick. Supporting polyploidy in genetic algorithms using dominance vectors. In *6th Intl. Conf. on Evolutionary Programming*, volume 1213 of LNCS, pages 223–234, Springer, 1997.

[6] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: fitness landscapes and GA performance. In *Proc. of the 1st European Conf. on Artificial Life*, pages 245–254, 1992.

[7] K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *6th Intl. Conf on Genetic Algorithms*, pages 159–166, 1955.

[8] T. Park and K. R. Ryu. A dual population genetic algorithm with evolving diversity. In *IEEE Congress on Evolutionary Computation (CEC2007)*, pages 3516–3522, 2007.

[9] T. Park, R. Choe, and K. R. Ryu. Adjusting population distance for the dual-population genetic algorithm. In *Australian Conference on Artificial Intelligence (AI 2007) (LNCS 4830)*, pages 171–180, 2007.

[10] S. Tsutsui, Y. Fujimoto, and A. Chosh. Forking genetic algorithms: GAs with search space division schemes. *Evolutionary Computation*, 5(1):61–80, 1997.

[11] F. Vavak, K. Jukes, and T. C. Fogarty. Learning the local search range for genetic optimisation in nonstationary environments. In *IEEE Intl. Conf. on Evolutionary Computation (ICEC'97)*, pages 355–360, 1997.

[12] L. D. Whitley. Fundamental principles of deception in genetic search. In *Foundations of Genetic Algorithms 1*, pages 221–241, 1991.

[13] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: on separability, population size, and convergence. In *Journal of Computing and Information Technology*, volume 7, pages 33–47, 1999.

[14] S. Yang. Non-stationary problem optimization using the primal-dual genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC2003)*, pages 2246–2253, 2003.

[15] S. Yang and X. Yao. Experimental study on population-based incremental learning algorithm for dynamic optimization problems, In *Soft Computing*, volume 9, pages 815–834, 2005.