Forking Genetic Algorithms: GAs with Search Space Division Schemes

Shigeyoshi Tsutsui
Department of Management
and Information Science
Hannan University
5-4-33 Amamihigashi, Matsubara
Osaka 580, Japan
tsutsui@hannan-u.ac.jp

Yoshiji Fujimoto
Department of Applied Mathematics
and Informatics
Faculty of Science and Technology
Ryukoku University
1-5 Yokoya, Seta Ooe, Ohtsu
Shiga 520-21, Japan
fujimoto@math.ryukoku.ac.jp

Ashish Ghosh
Machine Intelligence Unit
Indian Statistical Institute
203 B. T. Road
Calcutta 700035, India
ash@isical.ernet.in

Abstract

In this article, we propose a new type of genetic algorithm (GA), the forking GA (fGA), which divides the whole search space into subspaces, depending on the convergence status of the population and the solutions obtained so far. The fGA is intended to deal with multimodal problems that are difficult to solve using conventional GAs. We use a multi-population scheme that includes one parent population that explores one subspace and one or more child populations exploiting the other subspace. We consider two types of fGAs, depending on the method used to divide the search space. One is the *genotypic fGA* (g-fGA), which defines the search subspace for each subpopulation, depending on the *salient schema* within the genotypic search space. The other is the *phenotypic fGA* (p-fGA), which defines a search subspace by a *neighborhood hypercube* around the current best individual in the phenotypic feature space. Empirical results on complex function optimization problems show that both the g-fGA and the p-fGA perform well compared to conventional GAs. Two additional utilities of the p-fGA are also studied briefly.

Keywords

Forking genetic algorithm, search space division, multipopulation genetic algorithm, genotypic forking, phenotypic forking, salient schema, neighborhood hypercube.

1. Introduction

There are many genetic algorithm (GA)-hard problems that are difficult to solve by conventional GAs (Goldberg, 1989; Whitley, 1991). Several kinds of modified GAs have been developed to solve difficult problems. CHC (Eshelman, 1991) combines a conservative selection strategy that always preserves the best individuals found so far with a radical recombination operator that produces offspring that are maximally different from both parents.

Messy genetic algorithms (mGAs) (Goldberg, Korb, & Deb, 1989; Goldberg, Deb, & Korb, 1990) process variable-length strings that may be either underspecified or overspecified with respect to the problem being solved. GENITOR (Whitley, 1989) is specifically designed to allocate reproductive trials according to rank. Delta coding (Mathias & Whitley, 1994a,b) dynamically changes the representation of the search space in an attempt to exploit different problem representations. Finally, niche methods (Deb & Goldberg, 1989; Beasley, Bull, & Martin, 1993) extend the application of GAs to domains that require the location of multiple solutions.

In this article we propose a new type of GA, the *forking genetic algorithm* (fGA), which divides the whole search space into subspaces, depending on the status of convergence of the present population and the solutions obtained so far. The fGA (Tsutsui & Fujimoto, 1993, 1995) is also intended to deal with multimodal problems that are difficult to solve by the conventional GAs. We use a multipopulation scheme, which includes one parent population with a *blocking mode* (or exploration mode) and one or more child populations with a *shrinking mode* (or exploitation mode) generated by *population forking*. Each of these populations takes a different role in the optimization task; that is, different populations are responsible for searching nonoverlapping subareas in the search space.

Depending on the type of search space to be divided, we consider two types of fGAs. One is the *genotypic fGA* (g-fGA), which divides the genotypic search space, and the other is the *phenotypic fGA* (p-fGA), which uses the phenotypic parameter domain for space division. In the g-fGA, each population searches in a subspace defined by a *salient schema* in the genotypic search space. In the p-fGA, the corresponding subspace is defined by a *neighborhood hypercube* around the current best individual in the phenotypic parameter space. Empirical results show that both the g-fGA and the p-fGA yield good performance compared to conventional GAs; and the p-fGA shows an advantage over the g-fGA. We also discuss two other utilities of the p-fGA. One is the *variable resolution searching scheme* (vp-fGA) to solve multimodal problems with high precision. The other is the niche formation capability of the p-fGA.

In Section 2, we describe the basic model of evolution. Genotypic and phenotypic population forking schemes are then discussed in Section 3. In Section 4, we analyze the empirical results of the g-fGA and the p-fGA. Two utilities of the p-fGA are described in Section 5. Finally, concluding remarks are made in Section 6.

2. Basic Model of Evolution

Although the basic principles of fGAs do not depend on any special evolution model, in this work we used a modified evolution scheme, which shows better performance than conventional ones. The scheme basically involves applying *crossover* and *normal mutation* or *high mutation* followed by *population elitist selection* (Fig. 1). The scheme is described as follows. Let the size of the population $P(t-1)$ at generation $(t-1)$ be N. First, we copy this population to another pool $P'(t-1)$. We find out the canonical Hamming distance H_{ij} (where H_{ij} = (Hamming distance (S_i, S_j))$/L$; where L is the string length) of a crossover pair of individuals (S_i, S_j) in $P(t-1)$. Crossover of this pair is done with probability $(H_{ij})^{\alpha}$, where α $(0 < \alpha \leq 1)$ is called the *crossover Hamming power*; normal mutation with rate P_{nm} is applied after crossover. Offspring thus generated are stored in a population denoted by $C(t-1)$. When a decision not to do crossover is made, high mutation with rate P_{hm} ($P_{hm} \gg P_{nm}$) is applied to the parent individual with lower fitness so as to replace it in $P'(t-1)$. The best N individuals are then selected from the population $P'(t-1)$ and the offspring population

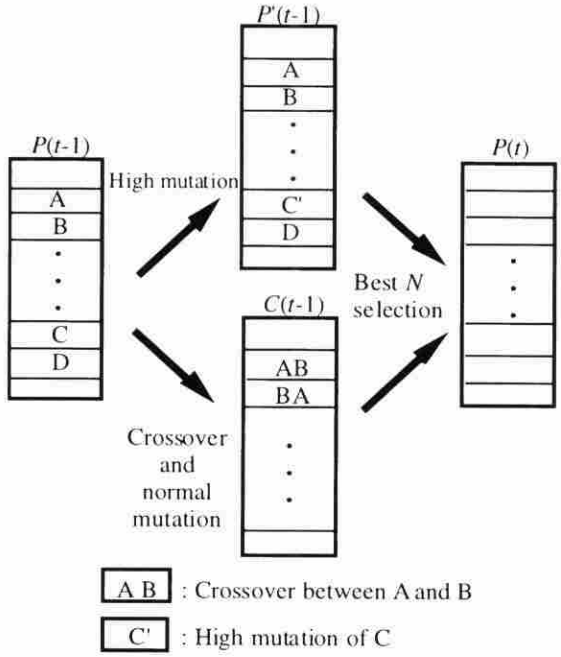

Figure 1. Basic model of evolution.

$C(t-1)$. This selection method is called *population elitist selection* (Eshelman, 1991), since it guarantees that the best N individuals seen so far always survived.

When the canonical Hamming distance H_{ij} between two individuals becomes small, probability $(H_{ij})^{\alpha}$ for crossover of this pair decreases (Fig. 2) and, consequently, high mutation is performed. Thus, an appropriate amount of diversity can be maintained in the population by a proper choice of α.

3. Population Forking

During the process of evolution, if the population is converged with reduced diversity, or the best solution obtained so far does not get updated for several consecutive generations, the process may be forked to allow searching concurrently in two different subpopulations. Thus, the whole search space is divided into subspaces, depending on the status of convergence of the present population and the solutions obtained so far. Search is continued independently in these subspaces. We call this method *population forking* (Fig. 3). These subpopulations are called the *parent population* and *child population*. Two techniques can be adopted for population forking: *genotypic population forking* and *phenotypic population forking*. Because the genotypic forking scheme provides better insight into the forking strategies, we describe it first.

3.1 Genotypic Forking GA

In this subsection, we describe the *genotypic forking GA* (g-fGA), which uses genotypic population forking for evolution. In the g-fGA, a search subspace is defined according to the *salient schema* in the genotypic search space.

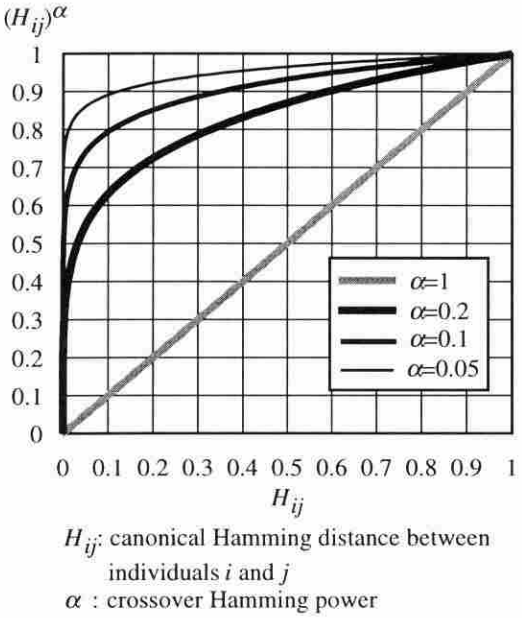

Figure 2. Relationship between H_{ij} and $(H_{ij})^{\alpha}$.

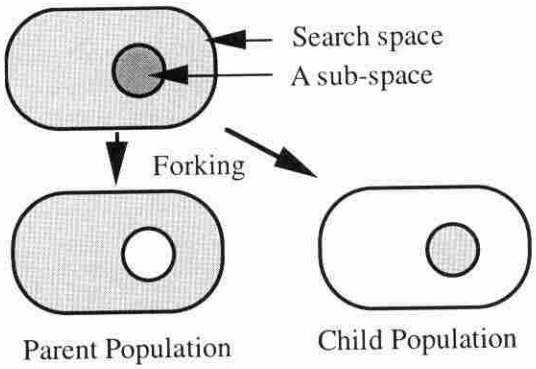

Figure 3. Population forking.

3.1.1 Salient Schema Let the population $P(t)$ be represented by the following matrix:

$$P(t) = (p_{ij}^t), \qquad i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, L \tag{1}$$

where p_{ij}^t is 0 or 1. Thus, each row vector represents the string of an individual of length L. To define a salient schema, we first introduce a concept of *temporal schema* ($TS(t)$) as a string of length L with elements 0, 1, and $*$ as follows:

$$TS(t) = (ts_1^t, ts_2^t, \ldots, ts_j^t, \ldots, ts_L^t)$$

$$
P(t) = \begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
$$

$$
TS(t) = (\ * \ \ 1 \ \ 0 \ \ 0 \ \ * \ \ 1 \ \ * \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ * \ \ 1 \ \ * \ \ * \ )
$$

Figure 4. Example of a temporal schema ($K_{TS} = 0.85$).

$$
\begin{aligned}
TS(t\text{-}4) &= (\ * \ \ 1 \ \ 0 \ \ 0 \ \ * \ \ 1 \ \ * \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ * \ \ 1 \ \ * \ \ * \ ) \\
TS(t\text{-}3) &= (\ 0 \ \ 1 \ \ 0 \ \ 0 \ \ * \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ * \ \ 1 \ \ 0 \ \ 0 \ ) \\
TS(t\text{-}2) &= (\ 0 \ \ 1 \ \ 0 \ \ 0 \ \ 1 \ \ 1 \ \ * \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ 0 \ \ 1 \ \ 0 \ \ 0 \ ) \\
TS(t\text{-}1) &= (\ 0 \ \ 1 \ \ 0 \ \ 0 \ \ * \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ 1 \ \ 0 \ \ * \ \ 1 \ \ 0 \ \ 0 \ ) \\
TS(t) &= (\ 0 \ \ 1 \ \ 0 \ \ 0 \ \ 1 \ \ 1 \ \ 0 \ \ * \ \ 0 \ \ 1 \ \ 0 \ \ * \ \ 1 \ \ * \ \ 0 \ ) \\
\hline
SS(t) &= (\ * \ \ 1 \ \ 0 \ \ 0 \ \ * \ \ 1 \ \ * \ \ * \ \ 0 \ \ 1 \ \ 0 \ \ * \ \ 1 \ \ * \ \ * \ )
\end{aligned}
$$

Figure 5. Example of a salient schema ($K_H = 5$).

$$
ts_j^t = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{N} p_{ij}^t \geq N \times K_{TS} \\ 0 & \text{if} \quad \sum_{i=1}^{N} (1 - p_{ij}^t) \geq N \times K_{TS} \\ * & \text{otherwise} \end{cases} \tag{2}
$$

where K_{TS} is a temporal schema detection threshold ($0.5 < K_{TS} \leq 1.0$). $TS(t)$ shows the state of convergence in each of the string positions. An example of a temporal schema is shown in Figure 4. Then, the *salient schema* $SS(t)$, a string of length L with elements 0, 1, and $*$, is defined from the temporal schemata as

$$
\begin{aligned}
SS(t) &= (ss_1^t, ss_2^t, \ldots, ss_j^t, \ldots, ss_L^t) \\
ss_j^t &= \begin{cases} 1 & \text{if} \quad ts_j^t = ts_j^{t-1} = \cdots = ts_j^{t-(K_H-1)} = 1 \\ 0 & \text{if} \quad ts_j^t = ts_j^{t-1} = \cdots = ts_j^{t-(K_H-1)} = 0 \\ * & \text{otherwise} \end{cases}
\end{aligned} \tag{3}
$$

where K_H is a salient schema detection constant ($K_H \geq 1$). $SS(t)$ is basically formed by robust building blocks in the temporal schema surviving through consecutive K_H generations. An example of a salient schema is shown in Figure 5.

The order of K_H can roughly be estimated from the *takeover time* t^* (Goldberg & Deb, 1991), which is the time required for a single individual to take over and occupy the whole

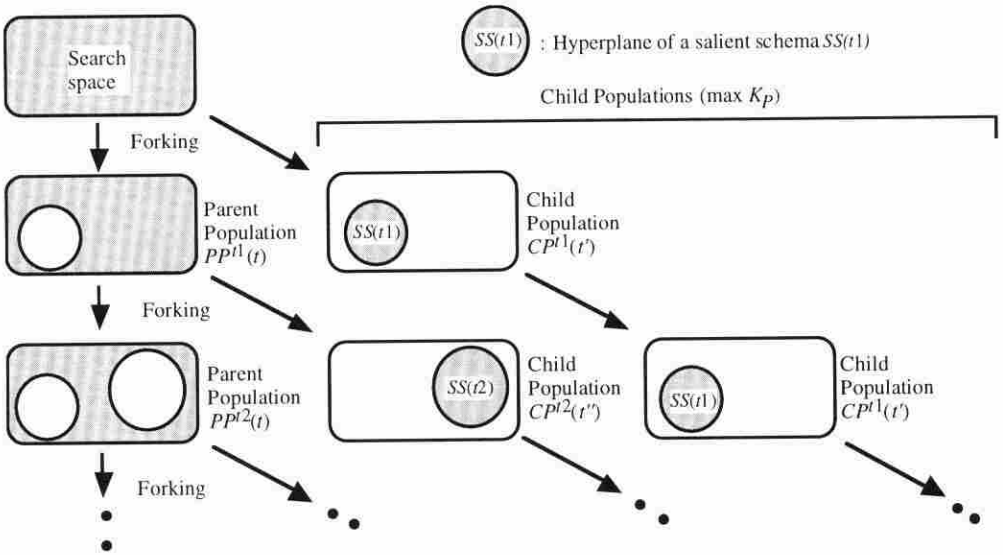

Figure 6. Genotypic population forking.

population under selection. For the selection scheme described in Section 2, the takeover time can be calculated as $t^* = \log_2 N$ when no other operator except selection is applied. K_H should be typically much larger than t^*.

3.1.2 Genotypic Population Forking During execution of a GA, sometimes the best solution obtained so far does not get updated for several consecutive generations. The current best solution may be a local optimum or the global optimum or close to some optimum. At this stage (formally described in Section 3.1.3), we make the initial population fork into a *parent population* $PP^{t1}(t)$ and a *child population* $CP^{t1}(t')$ covering the subspaces as shown in Figure 6.

After the population forking has occurred, individuals that are included in the salient schema will be deleted (as shown in Fig. 7) from the parent population $PP^{t1}(t)$, except that the best individual is retained. Next, $N - |PP^{t1}(t)|$ individuals are randomly regenerated outside the salient schema domain so as to keep the population size fixed. Thus, the diversity of $PP^{t1}(t)$ will be recovered, reducing the chance of being trapped in the local optima. Because

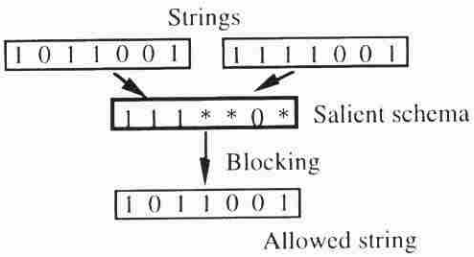

Figure 7. Blocking mode in the g-fGA.

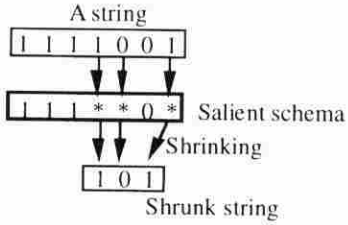

Figure 8. Shrinking mode in the g-fGA.

some of the individuals are blocked from entering $PP^{t1}(t)$ in the evolution, we sometimes call this the *blocking mode*.

In the population $CP^{t1}(t')$, all the individuals have the same value in the fixed positions of the salient schema $SS(t1)$. Values of "$*$" positions in the salient schema are gathered in the shrunk strings, as shown in Figure 8 (*shrinking mode*). Consequently, the size of the search space of $CP^{t1}(t')$ is reduced from 2^L to $2^{L'}$, where $L' = L - o(SS(t1))$, where $o(SS(t1))$ is the order of the salient schema $SS(t1)$. The subspace generated by the salient schema is then exploited with the shrunk strings to detect the best solution in that area.

Once again, detection of a new salient schema begins in the parent population. If the conditions for forking are satisfied, then the second child population is formed. A maximum of K_P ($K_P > 1$) child populations are allowed. The parent population and the child populations are evolved in *time-sharing* mode. Sharing of computation time by the parent and the child populations is defined by the BS_{ratio} on the generation counter. For example, ($BS_{\text{ratio}} = p : q$) means that we perform p generations for the parent population followed by q generations for each of the child populations; and this sequence continues.

Individuals are not exchanged between the child populations; but when an individual with the new best value is found in a child population, it is copied to the parent population. As a result, the best individual obtained so far is always included in the parent population. If the number of child populations is more than K_P, the oldest child population is discarded (see Fig. 6).

3.1.3 Conditions for Genotypic Forking The following three conditions are required simultaneously for genotypic population forking: (1) the best-so-far solution has not been updated for a specified number ($K_H > 1$) of generations; (2) the population has reduced diversity; and (3) the order of the salient schema is more than the specified constant $L \times K_o$, K_o ($0 < K_o < 1$) is the salient schema order threshold constant. The instants of population forking can be determined as:

$$t_F = \{t \mid (f_b(t - K_H + 1) = f_b(t) \wedge (B(t) \geq K_B) \wedge (o(SS^t) \geq L \times K_o)\} \tag{4}$$

where $f_b(t)$ is the best performance value up to generation t; $B(t)$ is the bias of population $P(t)$; K_B ($0.5 < K_B \leq 1.0$) is the bias threshold constant; and $\wedge$ is the "and" symbol. Here, bias $B(t)$ ($0.5 \leq B(t) \leq 1.0$) is a first-order convergence indicator showing the average percentage

of the prominent value in each position of the individuals (Grefenstette, Davis, & Cerys, 1991).

$$B(t) = \frac{1}{N \times L} \sum_{j=1}^{L} \left| \sum_{i=1}^{N} p_{ij}^{t} - \frac{N}{2} \right| + 0.5 \tag{5}$$

A large value of $B(t)$ means low genotypic diversity, and vice versa.

3.1.4 Salient Schema Composition When population forking occurs, a salient schema corresponding to the child population is maintained in a *salient schemata pool*. The new salient schema may be to combine with one of the old salient schemata maintained in the salient schemata pool so as to avoid duplicate subspaces in the child populations, or contiguous subspaces may be merged into a single subspace as described in (a) and (b). In this case, the old salient schema and the corresponding child population are discarded.

(a) Composition by Covering Relations If a new salient schema includes one of the existing salient schemata, the old salient schema is replaced by the new one. For example, the schema $SS(ti) = 1110{*}{*}{*}$ is included in the schema $SS(tj) = 111{*}{*}{*}{*}$. So, individuals blocked by schema $SS(ti) = 1110{*}{*}{*}$ are also blocked by the schema $SS(tj) = 111{*}{*}{*}{*}$. Thus, schema $SS(ti) = 1110{*}{*}{*}$ is discarded when the schema $111{*}{*}{*}{*}$ is detected. This avoids searching duplicate subspaces.

(b) Composition by Competitive Relation If a new salient schema $SS(tj)$ has the same order and differs in value only in one fixed position with one of the existing salient schema, then these two schemata are composed to a representative schema by replacing the different values of the fixed position with "$*$." Thus, the order of the schema is decreased by one, and two subspaces are merged to a single subspace. For example, schemata $111{*}{*}0{*}$ and $111{*}{*}1{*}$ are replaced by the single schema $111{*}{*}{*}{*}$.

3.2 Phenotypic Forking GA

The *phenotypic forking GA* (p-fGA) is described in this subsection. In the p-fGA, a subspace (child population) is defined by a *neighborhood hypercube* in the phenotypic search space around the current best solution.

Let the phenotypic parameter of a problem be $X = (x_1, x_2, \ldots, x_n)$. Let us consider a situation where there is no updating of the current best solution by a new individual for some consecutive generations. We represent the current best individual by its phenotypic parameter vector $X_C^t = (x_{1,c}^t, x_{2,c}^t, \ldots, x_{n,c}^t)$. Then, the neighborhood hypercube $R(X_C^t)$ around X_C^t may be defined as $R(X_C^t) = \{x_1, x_2, \ldots, x_n | (x_{i,c}^t - s_i/2) \leq x_i \leq (x_{i,c}^t + s_i/2)\}$, $i = 1, 2, \ldots, n$, where $S = (s_1, s_2, \ldots, s_n)$ defines the size of the neighborhood hypercube $R(X_C^t)$ and $s_i > 0$.

Conditions for the phenotypic population forking are as follows:

(i) The current best evaluated value has not been updated by a new individual for a specified number ($K_H \geq 1$) of generations; and

(ii) the number of the individuals located inside the neighborhood hypercube $R(X_C^t)$ is more than a specified number $N \times K_R (0 < K_R \leq 1.0)$, where K_R is the phenotypic convergence threshold constant.

If the conditions of forking are satisfied, we make the initial population fork into a parent population $PP^{t1}(t)$, which evolves outside $R(X_C^{t1})$, and a child population $CP^{t1}(t')$, which evolves inside $R(X_C^{t1})$, as shown in Figure 9. Condition (ii) shows that the population

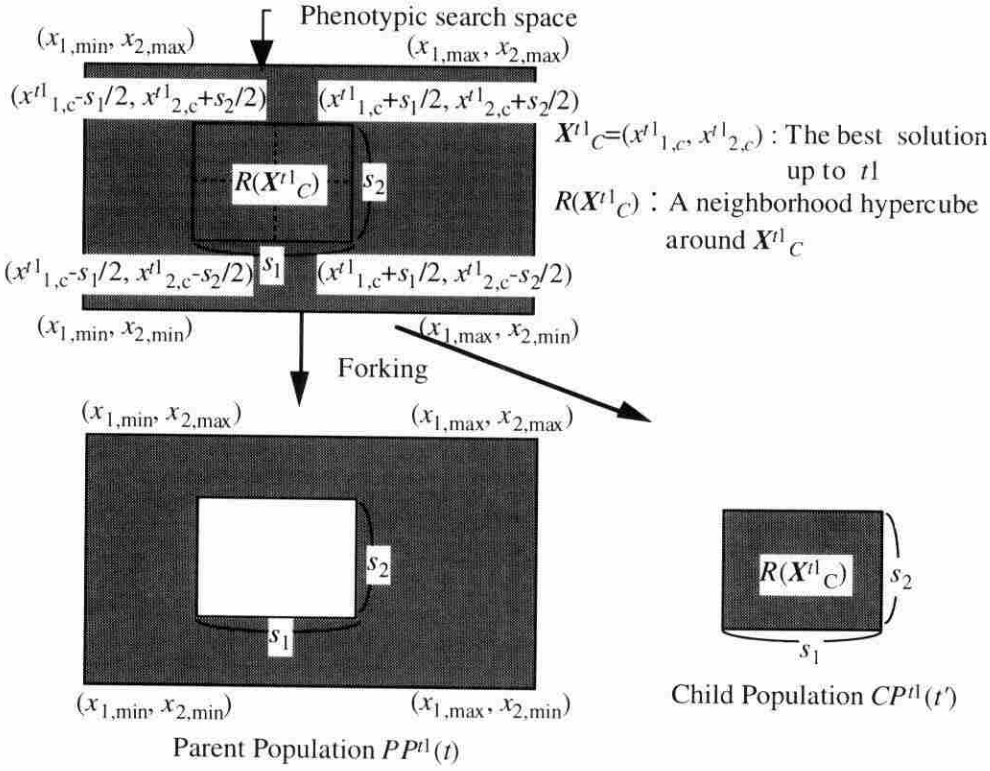

Figure 9. Phenotypic population forking.

has converged with reduced diversity [which in the g-fGA is modeled by conditions (b) and (c), Subsection 3.1.3].

After the population forking has occurred, individuals that are located inside $R(X_C^{t1})$, except the best individual, will be deleted from the parent population $PP^{t1}(t)$; individuals are randomly regenerated to keep the parent population size fixed. Thus, the diversity of $PP^{t1}(t)$ may be recovered so as to escape from being trapped in local optima as in the g-fGA. Figure 10 shows an example of the blocking mode in the p-fGA and corresponds to Figure 7 of the g-fGA. In this figure, there are two phenotypic parameters x_1 and x_2 in the range $0.0 \leq x_1, x_2 \leq 25.5$, which are coded by 8 bits. We assume $X_C^{t1} = (10.0, 6.0)$. Let the *precision* of parameter x_i be represented by Δx_i. Then, $\Delta x_1, \Delta x_2$ are both $0.1 (= (25.5 - 0.0)/(2^8 - 1))$. We consider the case where the parent and the child populations have the same precision. The *neighborhood hypercube size* S can be determined from the number of bits used to represent each of the parameters in the child population and the precision used. If 6 bits are used to represent both x_1 and x_2 in the child population, then S becomes $((2^6-1)\times 0.1, (2^6-1)\times 0.1) = (6.3, 6.3)$; and thus, $(10.0 - 3.2) \leq x_1 \leq (10.0 + 3.1)$ and $(6.0 - 3.2) \leq x_2 \leq (6.0 + 3.1)$, as shown in Figure 11. An individual with parameter values $x_1 = 10.1, x_2 = 5.1$, for example, is reencoded in $R(X_C^{t1})$ with 6 bits for each parameter; the total length of a string in the child population is 12. Thus, the search space of the child population is 1/16th $(= 2^{12}/2^{16})$ of the original search space.

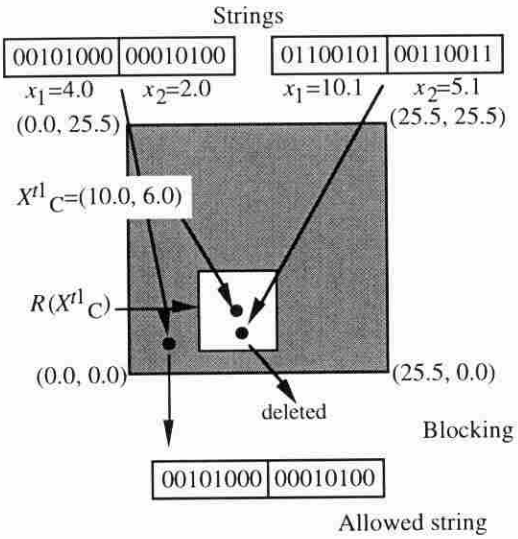

Figure 10. Blocking mode in the p-fGA.

The rest of this algorithm (i.e., how many times to fork and which child populations to discard) is similar to that of the g-fGA. We mention here that we did not do the *hypercube composition* as is done for the g-fGA.

4. Empirical Results

In this section, experimental results are analyzed to evaluate the g-fGA and the p-fGA. Two more GAs are tested. They are the nonforking GA (n-fGA) (population forking is not

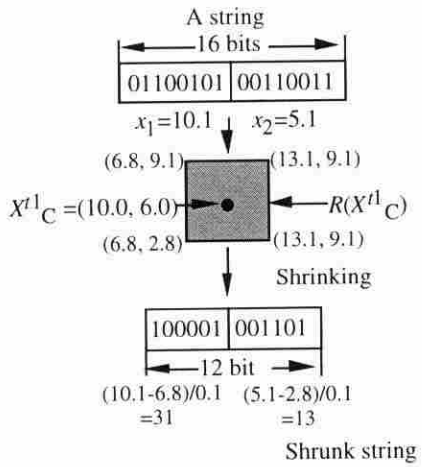

Figure 11. Shrinking mode in the p-fGA.

applied) and GENESIS (Grefenstette, Davis, & Cerys, 1991) with an elite option. The performance was tested on the following four test functions:

1. *De Jong's test function*: f_1. This is a well-known unimodal simple function (De Jong, 1975) defined as follows:

$$f_1 = \sum_{i=1}^{3} x_i^2 \tag{6}$$

Each parameter x_i is represented by a 10-bit Gray code in the range -5.12 to 5.11 with a precision of 0.01. The length of a string is $10 \times 3 = 30$ bits.

2. *Goldberg's deceptive function*: $f_{\text{deceptive}}$. This deceptive function is made up of 10 copies of a 3-bit fully deceptive function (Goldberg, Korb, & Deb, 1989) as follows:

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

We used a tightly coupled deceptive function in our experiment.

3. *Frequency modulation sound (FMS) parameter identification problem*: f_{fms} (Tsutsui & Fujimoto, 1993). Here, the problem is to specify six parameters $(a_1, w_1, a_2, w_2, a_3, w_3)$ of the FMS model represented by

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_3 \sin(w_3 t\theta))) \tag{7}$$

with $\theta = 2\pi/100$. The function f_{fms} is defined as the summation of square errors between the evolved data and the model data as follows:

$$f_{\text{fms}} = \sum_{t=0}^{100} (y(t) - y_0(t))^2 \tag{8}$$

where the model data are given by the following equation:

$$y_o(t) = 1.0 \times \sin(5.0t\theta - 1.5 \times \sin(4.8t\theta + 2.0 \times \sin(4.9t\theta))) \tag{9}$$

Each parameter is represented by an 8-bit Gray code in the range -6.4 to 6.35, and a precision of 0.05 is used. The total length of a string is $8 \times 6 = 48$ bits.

4. *Modified Griewank function*: f_{Griewank} (Törn & Žilmskas, 1989). The function is defined as follows:

$$f_{\text{Griewank}} = \sum_{i=1}^{5} x_i^2/4000 - \sum_{i=1}^{5} \cos(x_i/\sqrt{i}) + 1 \tag{10}$$

Each parameter x_i is represented by a 10-bit Gray code in the range -51.2 to 51.1 with a precision of 0.1. The total length of a string is $10 \times 5 = 50$ bits.

The maximum number of trials was set at 3,000, 10,000, 100,000, 140,000 for f_1, $f_{\text{deceptive}}$, f_{fms}, and f_{Griewank}, respectively. Thirty simulations were made for each experiment. Searching continued until the global optimum was found, or the maximum number of trials was reached. A population size $N = 50$, Hamming power $\alpha = 0.05$, $P_{bm} = 0.1$, and $K_{TS} = 0.8$ were used for all the experiments. Other control parameters were tuned, and the optimum set is listed in Table 1. Except for the mutation rate, we used the default parameter values for experiments with GENESIS; the mutation rate was tuned. A two-point crossover operator was applied. We evaluated the models by measuring their number of runs in which the

Table 1. The g-fGA versus the p-fGA.

GA		Function	f_1	$f_{\text{deceptive}}$	f_{fms}	f_{Griewank}
g-fGA	Parameters	P_{nm}	0.006	0.01	0.02	0.02
		K_B	0.5	0.7	0.7	0.8
		K_H	5	40	60	60
		K_O	0.3	0.3	0.4	0.3
		BS_{ratio}	1:1	3:1	3:1	3:1
		K_P	1	3	3	2
	#OPT		30	30	28	30
	MNT		1,325.0	4,283.6	35,834.9	57,691.9
	#OPT/C*		1	1	0	8
p-fGA	Parameters	P_{nm}	0.006	0.01	0.02	0.02
		K_R	0.7	0.2	0.8	0.3
		K_H	5	20	60	60
		C-bits†	5	2	5	7
		BS_{ratio}	1:1	5:1	3:1	3:1
		K_P	1	2	3	2
	#OPT		30	25	30	30
	MNT		1,253.0	5,332.8	22,621.8	43,599.4
	#OPT/C*		16	0	7	22
n-fGA	P_{nm}		0.006	0.01	0.02	0.02
	#OPT		30	11	10	3
	MNT		1,302.1	3,426.7	9,206.9	39,026.7
GENESIS	P_{nm}		0.007	0.01	0.01	0.01
	#OPT		3	5	8	6
	MNT		2,242.7	8,080.6	65,937.9	47,241.2

*Number of runs in which the optimal solution was found in one of the child populations. †Number of bits used for each parameter in child population.

algorithm succeeded in finding the global optimum (#OPTs) and mean number of trials to find the global optimum in those runs where it did find the optimum (MNTs). Figure 12 shows the #OPT for the restricted number of trials, and Table 1 summarizes the results after the maximum number of trials.

For the function f_1, the #OPTs of the g-fGA, p-fGA, and n-fGA were all 30. The MNTs of the g-fGA and p-fGA were nearly the same as that of the n-fGA. Thus, the performance of all these algorithms was similar for the function f_1. For the function $f_{\text{deceptive}}$, the g-fGA performed better (#OPT = 30, MNT = 4,283.6) than the p-fGA (#OPT = 25, MNT = 5,332.8). The n-fGA (#OPT = 11, MNT = 3,426.7) and GENESIS (#OPT = 5, MNT = 8,080.6) showed very poor performance. The performance of the g-fGA (#OPT = 28, MNT = 35,834.9) and p-fGA (#OPT = 30, MNT = 22,621.8) was reversed for the function f_{fms}; the p-fGA showed better results. As usual, the n-fGA and GENESIS performed poorly. Similar results were produced by the g-fGA (#OPT = 30, MNT = 57,691.9) and p-fGA (#OPT = 30, MNT = 43,599.4) for the function f_{Griewank}; the results of the n-fGA and GENESIS were again poor. An overall analysis shows that the g-fGA and p-fGA outperformed both the n-fGA and GENESIS.

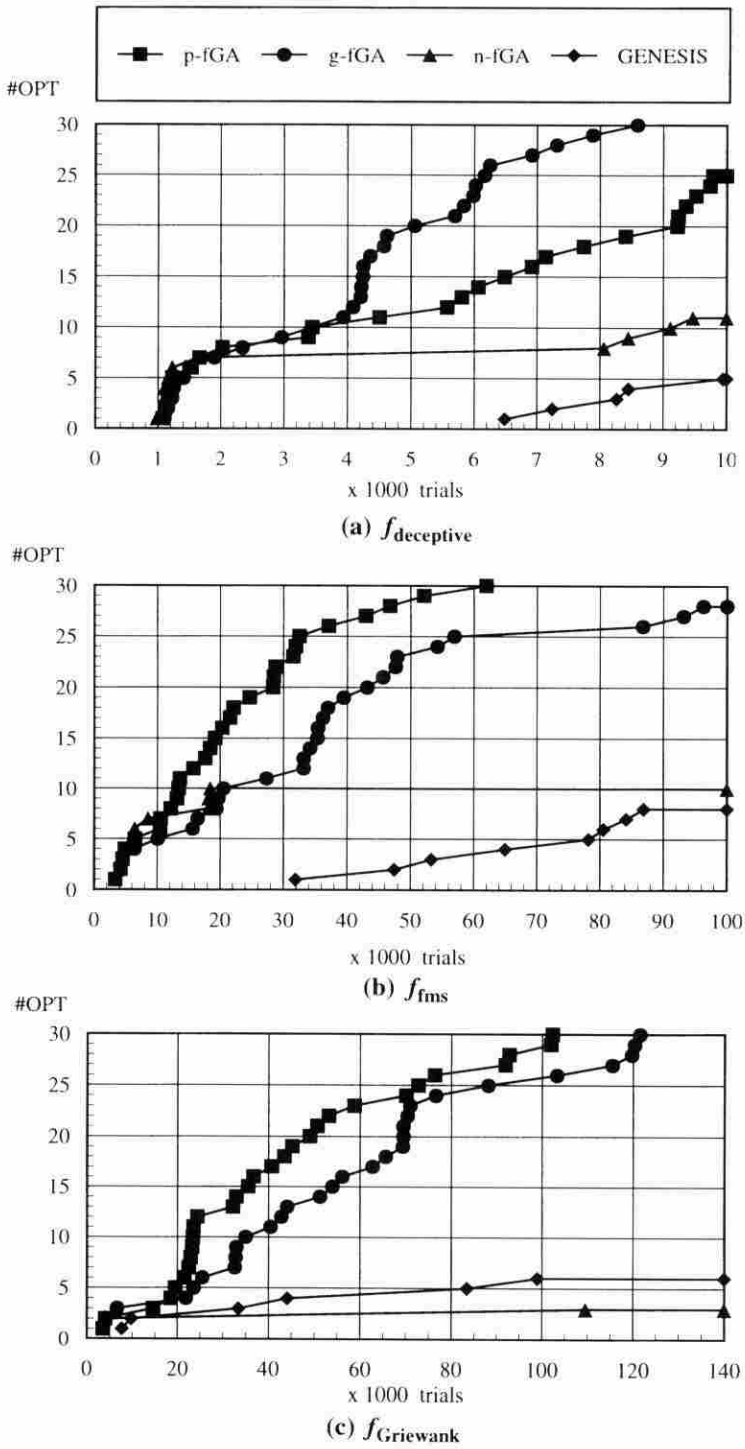

Figure 12. #OPT for restricted number of trials.

The performance of the g-fGA and p-fGA depended on the type of the problem under consideration. For the function $f_{\text{deceptive}}$, the optimum solution was always detected in the parent population by both the g-fGA and p-fGA. For the function f_{fms}, the g-fGA always detected the optimum in the parent population; whereas the p-fGA found the optimum seven times in the child populations. A similar trend was seen for the function f_{Griewank} (g-fGA detected the optimum solution eight times in the child populations, and the p-fGA did the same 22 times). Thus, for more complex functions (f_{fms} and f_{Griewank}), we see that the p-fGA maintained a balance in exploring the parent population and exploiting the child populations, which resulted in a better performance than that for the g-fGA. This is also evident from Figure 12 (b and c).

5. Two Other Utilities of Phenotypic Forking GAs

We have seen from the results in Section 4 that the p-fGA maintained a balance in exploring the parent population and exploiting the child populations and showed better performance than the g-fGA. In this section we consider two more utilities of the p-fGA; *variable-resolution searching* and *niche formation*.

5.1 Variable-Resolution p-fGA

The p-fGA described in Section 3.2 uses the same resolution Δx_i for the parent and the child populations. Hereafter, we call this p-fGA the *fixed resolution p-fGA* (fp-fGA). We may use different Δx_i values for the parent and the child populations. This type of GA may be called the *variable-resolution p-fGA* (vp-fGA). Thus, the vp-fGA provides more flexibility for defining the size of the neighborhood hypercube. Let us consider the case where we want to increase the size of the neighborhood hypercube with the fp-fGA, as shown in Figure 10. This can only be attained by increasing the number of bits to represent strings of the child population. However, if we increase 1 bit to represent x_1, then the value of s_1 increases from 6.3 to 12.7, thus almost doubling its size. In the vp-fGA, each Δx_i is recalculated for a given S and a given number of bits to represent members of the child population. Thus, we can take any value for S, although it may be that the string length of the members of the child populations becomes longer than that of the fp-fGA.

With the vp-fGA we basically can achieve variable resolution searching, as follows (Fig. 13):

1. The parent population is searched with a lower resolution and detects the near-optimal solution fast.
2. In the child populations, searching is performed with a higher resolution, depending on the problem, resulting in efficient detection of the global optimum or local optima, since searching proceeds in the smaller phenotypic space.

Variable resolution searching is similar to the *dynamic parameter encoding* (DPE) technique (Schraudolph & Belew, 1992), but the search space division scheme is completely different.

Next, let us evaluate the vp-fGA by comparing it with the fp-fGA. We use the two test functions f_{ripple} and $f_{\text{nonripple}}$, as follows:

$$f_{\text{ripple}} = \sum_{i=1}^{5} e^{-2\ln 2\left(\frac{x_i-0.1}{0.8}\right)^2} \left(\sin^6(5\pi x_i) + 0.1 \times \cos^2(500\pi x_i)\right) \tag{11}$$

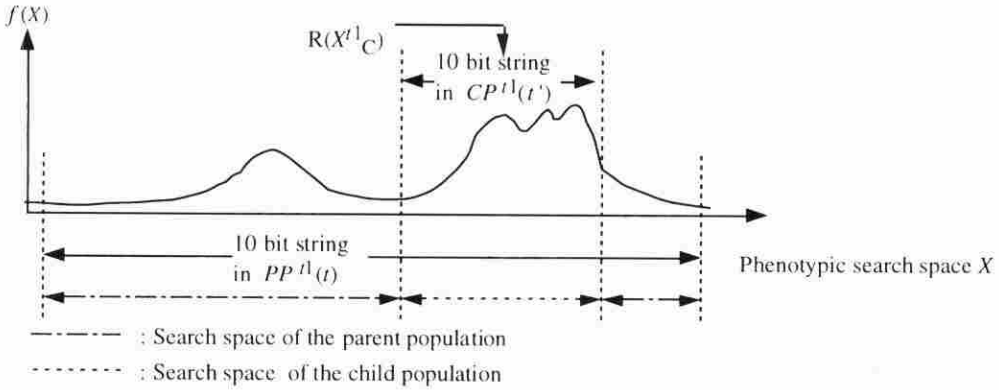

Figure 13. Variable resolution searching scheme of the vp-fGA.

$$f_{\text{nonripple}} = \sum_{i=1}^{5} e^{-2\ln 2\left(\frac{x_i - 0.1}{0.8}\right)^2} \sin^6(5\pi x_i) \tag{12}$$

where each x_i is in the range $0.0 \leq x_i \leq 100.0$, $i = 1, 2, \ldots, 5$. The function f_{ripple} has many main peaks of different sizes surrounded by a high frequency of small peaks. The function $f_{\text{nonripple}}$ does not have a high frequency of small peaks. Both of these functions have the maximum value at $x_1 = x_2 = , \ldots, x_5 = 0.1$, with functional value 5.5. We choose these functions because they require very high resolution to detect the actual optima. Assume that the problem is to find the optimal point with a resolution of 0.0001 for each x_i. Thus, we assume that the GA is able to find the optimal solution if the parameters $x_1, x_2, \ldots, x_5$ of the best individual are within the range $[(0.1 - 0.0001), (0.1 + 0.0001)]$.

The following experimental conditions are used. Thirty runs are performed. Each run continues until the global optimum is found or a maximum of 100,000 trials is reached. A population size of 50, Gray coding, and two-point crossover are used. Other parameters are tuned in the ranges $K_R \in [0.5, 0.7]$, $K_H = \{60, 100\}$, $K_P \in [1, 3]$, $P_{nm} \in [0.006, 0.02]$, $BS_{\text{ratio}} \in [1:1, 3:1]$, and a total of 48 combinations are tried so that the #OPT of the fp-fGA for function f_{ripple} is maximum. The size of the neighborhood hypercube (S) was set close to the diameter (=0.15) of the main peaks.

In the vp-fGA, we use $s_i = 0.15$ for all i. To represent each parameter x_i, 12 and 11 bits are used in the parent and child populations, respectively. Thus, the resolution Δx_i of the parent and child populations are $0.02442 (= 100.0/(2^{12} - 1))$ and $0.0000723 (= 0.15/(2^{11} - 1))$, respectively. In the fp-fGA, each x_i used 20 and 11 bits for its representation in the parent and child populations, respectively. Thus, the resolution Δx_i of the parent and child populations is $0.0000953 (= 100.0/(2^{20} - 1))$, and $s_i = 0.195091 (= 0.0000953(2^{11} - 1))$.

Simulation results are shown in Table 2. For the function $f_{\text{nonripple}}$, the results of the fp-fGA and vp-fGA are nearly the same; the #OPTs of the fp-fGA and vp-fGA were both 30 (100%), and the MNTs of the fp-fGA and vp-fGA were 16,845.7 and 22,953.1, respectively. For the function f_{ripple}, vp-fGA showed better performance; the #OPT of the vp-fGA was 30 (100%) and that of the fp-fGA was only 14 (47%), the MNTs of the fp-fGA and vp-fGA were 65,272.9 and 21,087.4, respectively. The n-fGA and GENESIS could not find the global optimum in any of the 30 runs for these functions. Figure 14 shows the #OPT for the restricted number of trials for both f_{ripple} and $f_{\text{nonripple}}$ and confirms these results.

Table 2. The fp-fGA versus the vp-fGA.

GA		fp-fGA	vp-fGA
String length	Parent Population	100(=20 * 5) bits	60(=12 * 5) bits
	Child Population	55(=11 * 5) bits	55(=11 * 5) bits
Size of neighborhood hypercube s_i		0.1950791	0.15
Resolution Δx_i	Parent Population	0.0000953	0.02442
	Child Population	0.000953*	0.0000723
Other parameters		$K_R = 0.7$, $K_H = 100$, $K_P = 2$, $P_{nm} = 0.02$, $BS_{\text{ratio}} = 2{:}1$	
$f_{\text{nonripple}}$	#OPT	30	30
	#OPT/P*	2	—
	#OPT/C†	28	30
	MNT	16,845.7	20,916.0
f_{ripple}	#OPT	14	30
	#OPT/P*	4	—
	#OPT/C†	10	30
	MNT	65,272.9	21,087.4

*Number of runs in which the optimal solution was found in the parent population. †Number of runs in which the optimal solution was found in one of the child populations.

Thus, it is evident that the capability of the vp-fGA for finding the global optimum with high resolution is fairly good. With this feature of the vp-fGA, we can compensate for the lack of local search capability of GAs.

5.2 Niche Formation Feature of p-fGA

We take a simple multimodal function to show that the p-fGA has the niche-forming capability:

$$f_{\text{niche}} = e^{-2\ln 2\left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x) \tag{13}$$

This function was used by Deb and Goldberg (1989) to test their *sharing scheme*, and it has five peaks of different sizes. We used the fp-fGA, and the parameters for its simulation are as follows: maximum number of trials for both the parent and child populations = 1000, population size = 10, string length (Gray coded) = 30 (in the parent population), crossover Hamming power $\alpha = 0.3$, normal mutation rate $P_{nm} = 0.006$, high mutation rate $P_{bm} = 0.06$, maximum number of child populations $K_P = 5$, and the forking condition constant $K_R = 0.7$ and $K_H = 3$. The BS_{ratio} is set as 1:1. The string length of child population is set to 27. The neighborhood hypercube size (S) is then $(1.0/(2^{30} - 1))(2^{27} - 1) \sim 0.125$. This value is close to $\sigma_{\text{share}} \times 2$, where σ_{share} is a sharing parameter defined by Deb and Goldberg (1989). Figure 15 shows a typical process of the niche formation. With five child populations, all peaks are fairly well covered with solutions. We did 30 runs for this problem. All these

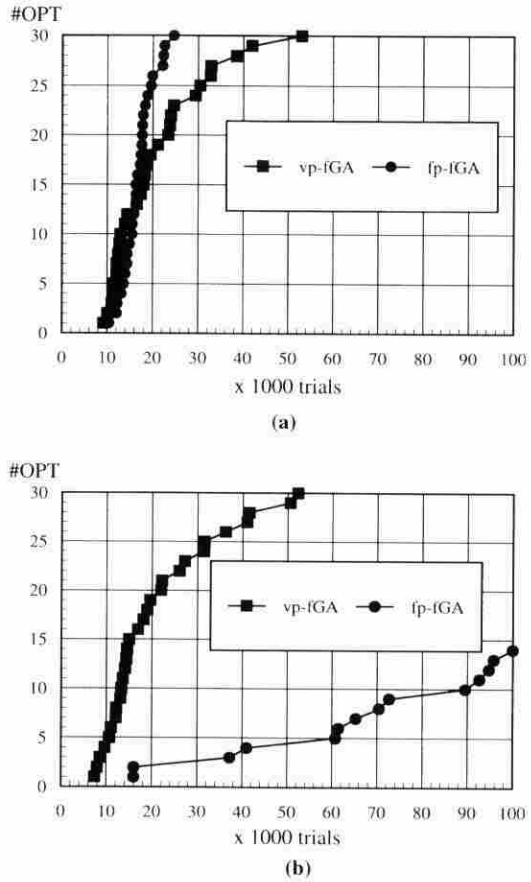

Figure 14. #OPT for restricted number of trials: (a) $f_{\text{nonripple}}$; (b) f_{ripple}.

experiments showed success in niche formation, although the sequences of covering the peaks were different for different runs. Similar results were also obtained with the vp-fGA. Parameters were chosen so as to ensure that a number of forkings occur.

6. Conclusions

In this article, we proposed a new type of GA, the forking GA (fGA) which is intended to deal with multimodal problems. We used a multipopulation scheme that includes one parent population that explores one subspace and one or more child population(s) that exploit the other subspace. We considered two types of fGAs, depending on the type of the search space to be divided. One is the genotypic fGA (g-fGA), which defines the search subspace for each population, depending on the salient schema within the genotypic search space. The other is the phenotypic fGA (p-fGA), which defines a search subspace by a neighborhood hypercube around the current best individual in the phenotypic feature space.

The empirical results on some complex function optimization problems showed that both the g-fGA and p-fGA perform fairly well compared with conventional GAs. Although performance of the g-fGA and p-fGA depends on the type of problem, for more complex

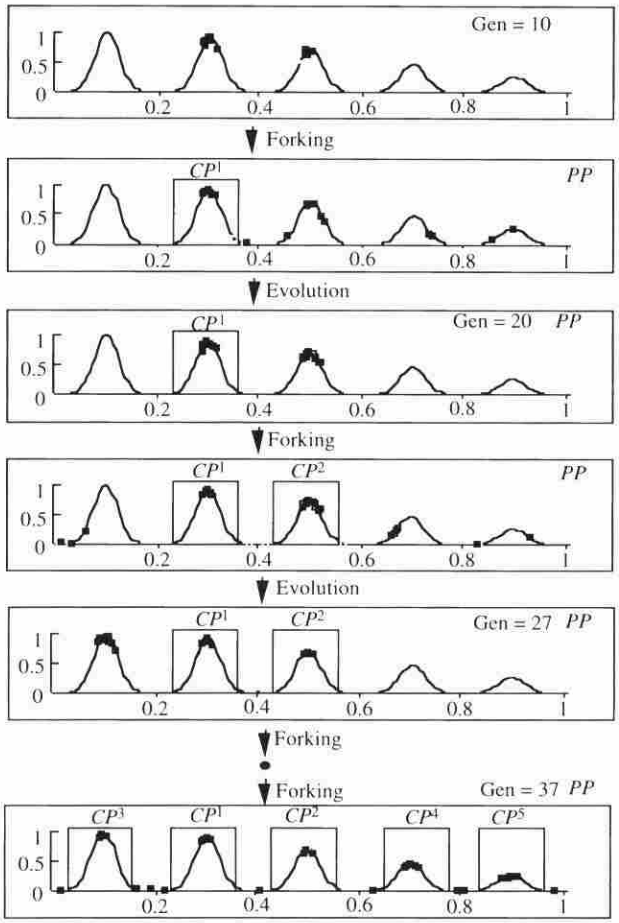

Figure 15. Example of niche formation by the p-fGA. Gen = generation.

functions we saw that the p-fGA maintains a balance in exploring the parent population and exploiting the child populations and showed better performance than the g-fGA. That the use of phenotypic search space information outperforms the use of genotypic search space information corroborates the earlier finding of Deb and Goldberg (1989) on sharing, where phenotypic sharing maintained better niching than did genotypic sharing. Two additional utilities of the p-fGA were also briefly studied. Although the p-fGA generally performs better than the g-fGA, the g-fGA will be useful for problems where genotypic feature directly maps to fitness.

There are many opportunities for further research related to the proposed technique: analyzing the extra overhead required for blocking and shrinking modes, studying the load balancing between the parent and child populations, and devising a more efficient method of discarding some of the child populations. Evaluating the effectiveness of the fGAs on real-life problems; comparing them with other multipopulation-based schemes; extending them for permutation problems and other evolution schemes, such as real coded GAs; and determining the optimal parameter set for the fGAs also remain to be investigated. Comparison of the variable resolution search capability of the p-fGA with the dynamic parameter

encoding technique and its niche formation capability with that of other niche techniques will constitute another part of future study. Furthermore, analytical analysis may establish some relation of the p-fGA with the global random searching technique (Peck & Dhawan, 1995).

Acknowledgments

We thank the reviewers for their stimulating comments. This research is partially supported by the Ministry of Education, Science, Sports and Culture, Japan, under Grant-in-Aid 264-08233105 for Scientific Research on Priority Areas.

References

Beasley, D., Bull, D. R., & Martin, R. R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation, 1(2)*, 101–125.

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 42–50). San Mateo, CA: Morgan Kaufmann.

De Jong, K. (1975). *Analysis of the behavior of a class of genetic adaptive systems.* PhD dissertation. Department of Computer and Communication Sciences. University of Michigan, Ann Arbor.

Eshelman, L. J. (1991). The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 265–283). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 69–93). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E., Korb, B., & Deb, K. (1989). Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems, 3*, 493–530.

Goldberg, D. E., Deb, K., & Korb, B. (1990). Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems, 4*, 415–444.

Grefenstette, J. J., Davis, L., & Cerys, D. (1991). *GENESIS and OOGA: Two GA systems.* Melrose, MA: TSP Publications.

Mathias, K., & Whitley, D. (1994a). Initial performance comparisons for the delta coding algorithm. In J. D. Schaffer (Ed.), *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 433–438). Piscataway, NJ: IEEE Press.

Mathias, K., & Whitley, D. (1994b). Changing representations during search: A comparative study of delta coding. *Evolutionary Computation, 2(3)*, 249–278.

Peck, C. C., & Dhawan, A. P. (1995). Genetic algorithms as global random search methods: An alternative perspective. *Evolutionary Computation, 3(1)*, 39–80.

Schraudolph, N. N., & Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning, 9*, 9–21.

Törn, A., & Žilmskas, A. (1989). Methods of generalized descent. In G. Goos & J. Hartmanis (Eds.), *Global optimization* (p. 186). Berlin: Springer-Verlag (Lecture Notes in Computer Science).

Tsutsui, S., & Fujimoto, Y. (1993). Forking genetic algorithm with blocking and shrinking modes. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 206–213).

San Mateo, CA: Morgan Kaufmann.

Tsutsui, S., & Fujimoto, Y. (1995). Phenotypic forking genetic algorithms. In D. Fogel (Ed.), *Proceedings of the IEEE International Conference on Evolutionary Computation* (pp. 566–572). Piscataway, NJ: IEEE Press.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 116–121). San Mateo, CA: Morgan Kaufmann.

Whitley, D. (1991). Fundamental principles of deception in genetic search. In G. J. E. Rawlins (Ed.), *Foundations of genetic algorithms* (pp. 221–241). San Mateo, CA: Morgan Kaufmann.