Evolutionary Computation, MIT Press, Vol. 5, No. 1, pp. 61-80

Forking GAs: GAs with Search Space Division Schemes

Abstract

In this article, we propose a new type of GA, the *forking GA (fGA)* which divides the whole search space into sub-spaces depending on the convergence status of the population and the solutions obtained so far. The fGA is intended to deal with multi-modal problems which are difficult to be solved by the conventional GAs. We use a multi-population scheme, which includes one parent population that explores one sub-space; and one or more child population(s) exploiting the other subspace. We consider two types of fGAs depending on the way of dividing the search space. One of them is the g-fGA (*genotypic fGA*) which defines the search sub-space for each sub-population depending on the *salient schema* within the genotypic search space. The other is the p-fGA (*phenotypic fGA*) which defines a search sub-space 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 fairly well over the conventional GAs. Two additional utilities of the p-fGA are also studied briefly.

Keywords

Forking GA, search space division, multi-population GA, genotypic forking, phenotypic forking, salient schema, neighborhood hypercube

----

Shigeyoshi Tsutsui: Department of Management and Information Science, Hannan University, 5-4-33 Amamihigashi, Matsubara, Osaka 580, JAPAN

Phone: +81-723-32-1224, Fax: +81-723-36-2633, E-mail: 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

Phone: +81-775-43-7488, Fax: +81-775-43-7524, E-mail: fujimoto@math.ryukoku.ac.jp

Ashish Ghosh: Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Calcutta 700035, INDIA

Phone: +91-33-552-8085 (ext 3110) Fax: +91-33-556-6680, E-mail: ash@isical.ernet.in

1. Introduction

There are many *GA-hard* problems that are difficult to be solved by the conventional GAs (Goldberg, 1989; Whitley,1991). Many kinds of modified GAs, such as CHC (Eshelman, 1991) which 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 of the parents; mGA (Goldberg, Korb & Deb, 1989; Goldberg, Deb & Korb,1990) which processes variable-length strings that may be either under or over-specified with respect to the problem being solved ; GENITOR (Whitley, 1989) which is specifically designed to allocate reproductive trials according to rank; delta coding (Mathias & Whitely, 1994a, 1994b) that dynamically changes the representation of the search space in an attempt to exploit different problem representations; and niche methods (Deb & Goldberg, 1989; Beasley, Bull & Martin, 1993) which extend the application of GAs to domains that require the location of multiple solutions, aimed to solve these problems are proposed in the literature.

In this article we propose a new type of GA, the *forking GA (fGA)*, which divides the whole search space into sub-spaces depending on the status of convergence of the present population and the solutions obtained so far. The fGA (Tsutsui & Fujimoto, 1993; Tsutsui & Fujimoto,1995) is also intended to deal with multi-modal problems which are difficult to be solved by the conventional GAs. We use a multi-population 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; i. e., different population is responsible for searching non-overlapping sub-areas in the search space.

Depending on the type of the 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 phenotypic parameter domain for space division. In the g-fGA, each population searches in a sub-space defined by a *salient schema* in the genotypic search space. In the p-fGA, the corresponding sub-space 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 show fairly good performance over the conventional GAs; and the p-fGA shows an upper edge over the g-fGA. We also discuss two other utilities of the p-fGA. One of them is the *variable resolution searching scheme* (*vp-fGA*) to solve multi-modal problems with high precision. The other is the niche formation capability of the p-fGA.

The rest of this articles is organized as follows. In Section 2, we describe the basic model of evolution. Genotypic and phenotypic population forking schemes are then discussed in Sections 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 compared to the conventional ones. The scheme basically involves applying *crossover & normal mutation* or *high mutation* followed by *population elitist selection*; and is shown in Figure 1. The scheme is described as follows. Let the size of the population $P(t\text{-}1)$ at generation $(t\text{-}1)$ be N. First we copy this population to another pool $P'(t\text{-}1)$. We find out the canonical Hamming distance H_{ij} (= Hamming distance (S_i, S_j) $/L$; L: length of an individual, S_i: an individual) of a crossover pair (S_i, S_j) in $P(t\text{-}1)$. 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 is stored in $C(t\text{-}1)$. When a decision of not doing crossover is made, high mutation with rate P_{hm} $(P_{hm} >> P_{nm})$ is applied on the individual with lower functional value so as to replace it in $P'(t\text{-}1)$. The best N individuals are then selected from the population $P'(t\text{-}1)$ and the offspring population $C(t\text{-}1)$. This selection

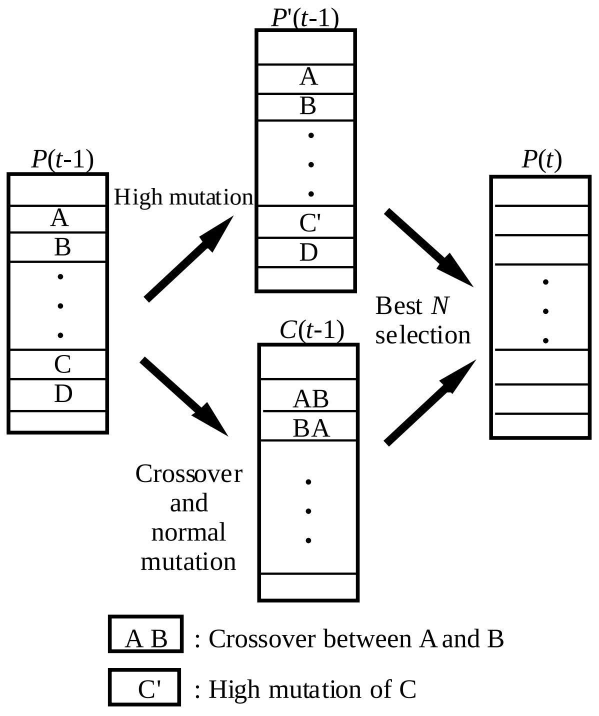


Figure 1. Basic model of evolution

method is called *population elitist selection* (Eshelman, 1991), since it guarantees that the best N individuals, seen so far, are always survived.

When the canonical Hamming distance H_{ij} between two individuals becomes small, probability $(H_{ij})^{\alpha}$ for crossover of this pair decreases (please see 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.

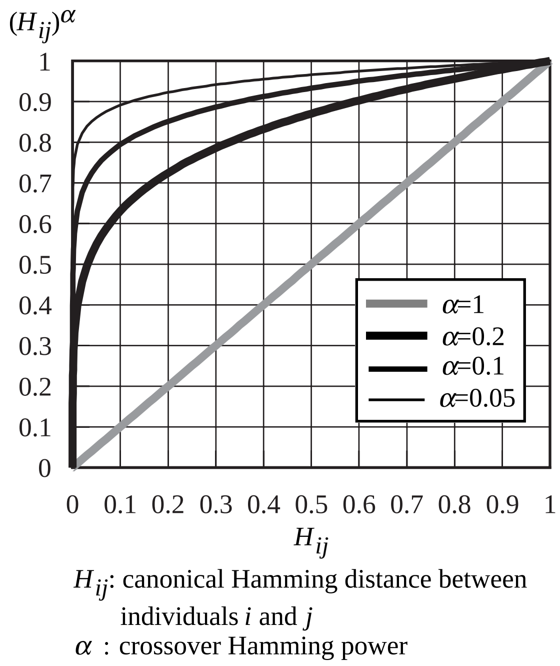


Figure 2. Relationship between H_{ij} and $(H_{ij})^{\alpha}$

3. Population Forking

During the process of evolution, if the population is converged to a smaller diversity or the best solution obtained so far does not get updated for some consecutive generations, the process may be forked to allow searching concurrently in two different sub-populations. Thus the whole search space is divided into sub-spaces depending on the status of convergence of the present population and the solutions obtained so far; and search is continued independently in these sub-spaces. We call this method *population forking* (please see Figure 3). These sub-populations are called *parent population* and *child population*. Two techniques can be adopted for population forking. These are *genotypic population forking* and *phenotypic population forking*. Since in the genotypic forking scheme provides a better insight of the forking strategies, we describe it first; and it is then followed by the phenotypic population forking.

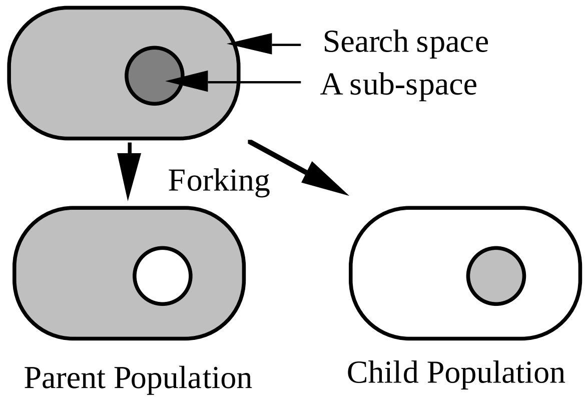


Figure 3. Population forking

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 sub-space is defined depending on the *salient schema* in the genotypic search space. So, let us first introduce its definition.

3.1.1 Salient Schema

Let us represent the population $P(t)$ by the following matrix.

$$P(t) = \left(\ p_{ij}^{t}\ \right), \quad i = 1, 2, \cdots, N, \quad j = 1, 2, \cdots, L\ ; \tag{1}$$

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" & "*" as follows:

$$\begin{aligned} TS(t) &= \left(ts_1^t, ts_2^t, \cdots, ts_j^t, \cdots, ts_L^t\right), \\ ts_j^t &= \begin{cases} 1 & \text{if } \sum_{i=1}^{N} p_{ij}^{t} \geq N \times K_{TS} \\ 0 & \text{if } \sum_{i=1}^{N} (1 - p_{ij}^{t}) \geq N \times K_{TS} \\ * & \text{otherwise;} \end{cases} \end{aligned} \tag{2}$$

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" & "*", is defined from the temporal schemata as

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

where K_H is called 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, D. E. & Deb, K. 1991) which is the time to occupy the whole population by a single individual. 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 it so happens that the best solution obtained so far does not get updated for some consecutive generations. The current best solution may be one of the local optima or the global optimum, or close to them. 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 sub-spaces as shown in Figure 6.

After the population forking has occurred, individuals which are included in the salient schema, except the best individual, will be deleted (as shown in Figure 7) from the parent population $PP^{t1}(t)$, and 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, having less chance of being trapped in the local optima. As some of the individuals are blocked to enter $PP^{t1}(t)$ in the evolution, we sometimes call this *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 $L' = L - o(\ SS(t1)\ )$; $o(\ SS(t1)\ )$ is the order of the salient schema $SS(t1)$. The sub-space 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

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

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

Figure 4. An example of a temporal schema (K_{TS}=0.85)

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

Figure 5. An example of a salient schema(K_H=5)

for forking is satisfied, then the second child population is formed. A maximum of $K_P\,(K_P \geq 1)$ child populations are allowed. The parent population and the child populations are evolved in *time sharing* mode. Sharing of computation time by the two populations is defined by the BS_{ratio} on the generation counter. For example, the $BS_{\text{ratio}} = p : q$ means 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 (please see Figure 6).

3.1.3 Conditions for Genotypic Forking

Following three conditions are required simultaneously for genotypic population forking:

(a) the best so far evolved value has not been updated for a specified number ($K_H \geq 1$) of generation,

(b) the population converges to a smaller diversity, and

(c) 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.

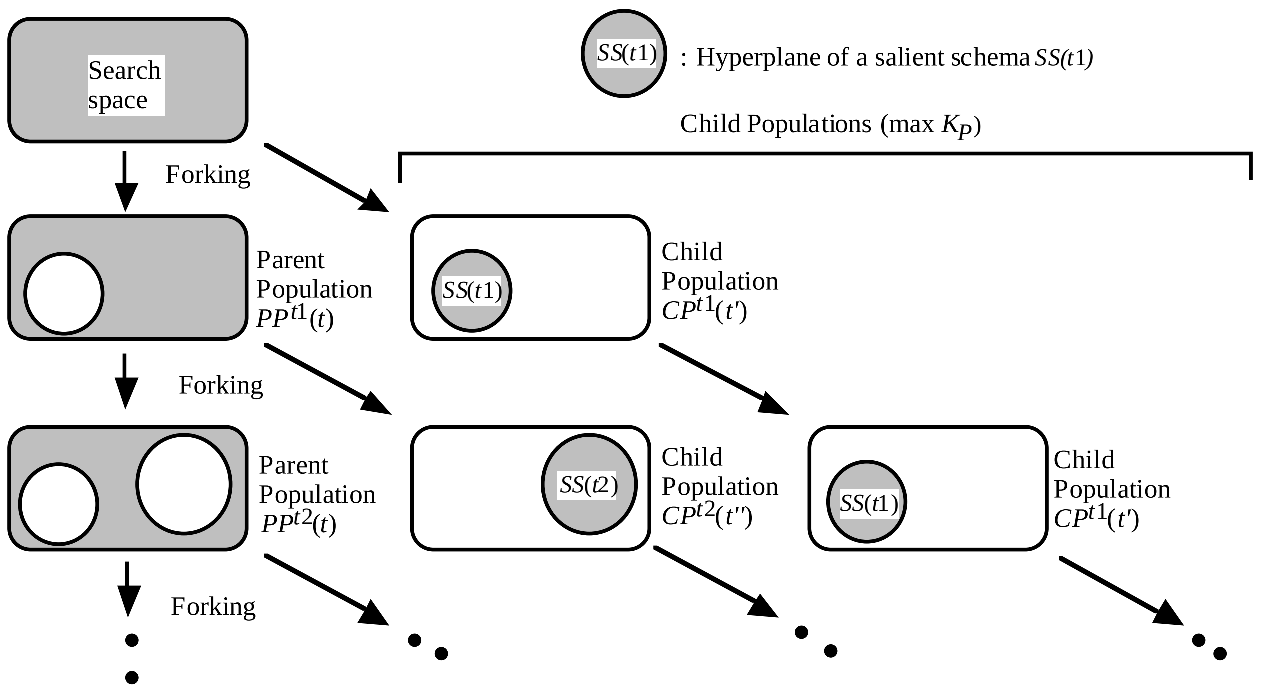
The instants of population forking can be determined as:

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

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 the symbol $\wedge$ means *and*. 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, and is defined as (Grefenstette, Davis & Cerys,1991)

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

Large value of $B(t)$ means low genotypic diversity and vice versa.



SS(t1) : Hyperplane of a salient schema SS(t1)
Child Populations (max K_P)
Search space
Forking
Parent Population $PP^{t1}(t)$
SS(t1)
Child Population $CP^{t1}(t')$
Forking
Parent Population $PP^{t2}(t)$
SS(t2)
Child Population $CP^{t2}(t'')$
SS(t1)
Child Population $CP^{t1}(t')$
Forking

Figure 6. Genotypic population forking

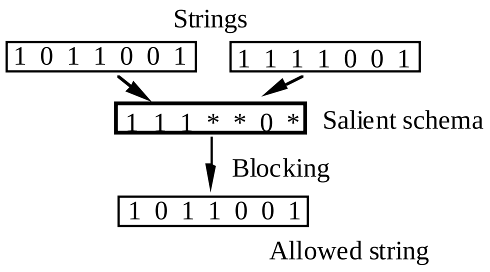


Figure 7. Blocking mode in the g-fGA

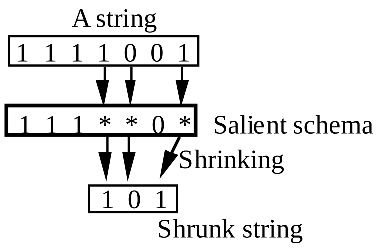


Figure 8. Shrinking mode in the g-fGA

3.1.4 Salient Schema Composition

When a population forking occurs salient schema corresponding to the child population is maintained in a *salient schemata pool*. The new salient schema is tried to combine with one of the old salient schemata maintained in the salient schemata pool so as to avoid duplicate sub-spaces in the child populations or merge contiguous sub-spaces into a single sub-space 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 included 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 sub-spaces.

(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 is 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 sub-spaces are merged to a single sub-space. For example, schemata 111**0* and 111**1* are replaced by the single schema 111****.

3. 2 Phenotypic Forking GA

Phenotypic forking GA (p-fGA) which uses phenotypic population forking 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 $X = (x_1, x_2, ..., x_n)$. 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 $X^t_C = (x^t_{1,c}, x^t_{2,c}, ..., x^t_{n,c})$. Then, the neighborhood hypercube $R(X^t_C)$ around X^t_C may be defined as $R(X^t_C) = \{x_1, x_2, ..., x_n \mid (x^t_{i,c} - s_i/2) \leq x_i \leq (x^t_{i,c} + s_i/2)$, i=1,2, ..., $n\}$, where $S =(s_1, s_2, ..., s_n)$ defines the size of the neighborhood hypercube $R(X^t_C)$ and $s_i > 0$.

Conditions for the phenotypic population forking are as follows:

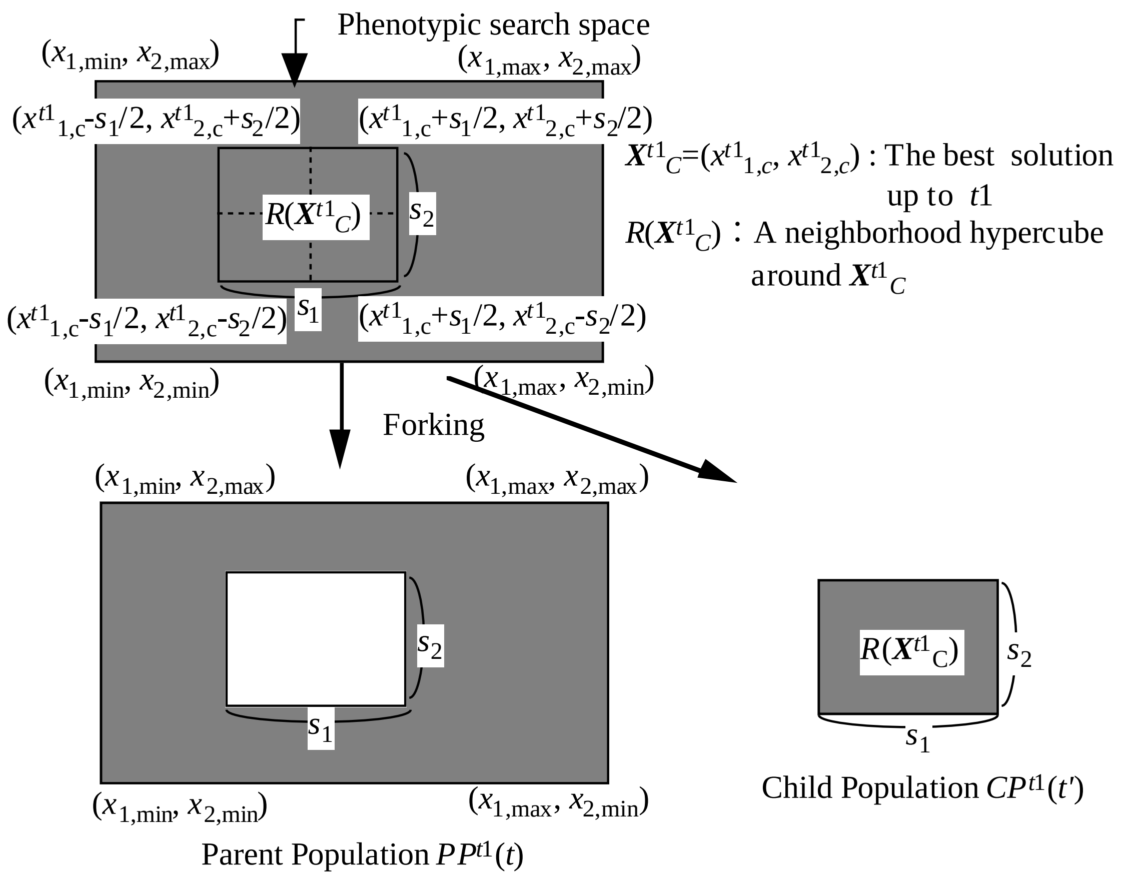
(i) the current best evaluated value has not been updated by a new individual for a specified number ($K_H \geq 1$) of generations, and

(ii) the number of the individuals located inside the neighborhood hypercube $R(X^t_C)$ is more than a specified number $N \times K_R$ ($0 < K_R \leq 1.0$), 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 $R(X^{t1}_C)$, and a child population $CP^{t1}(t')$ which evolves inside $R(X^{t1}_C)$ as shown in Figure 9. Condition (ii) shows that the population has converged to a smaller diversity (which in the g-fGA is modeled by conditions (b) & (c), Subsection 3.1.3).

After the population forking has occurred, individuals which are located inside $R(X^{t1}_C)$, 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 & x_2 in the range $0.0 \leq x_1, x_2 \leq 25.5$ and coded by 8 bits. We assume X^{t1}_C = (10.0, 6.0). Let the *precision* of parameter x_i be represented by Δx_i. Then Δx_1, Δx_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 six bits are used to represent both x_1 & 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 $(10.0-3.2) \leq x_1 \leq (10.0+3.1)$ & $(6.0-3.2) \leq x_2 \leq (6.0+3.1)$ as shown in Figure 11. An individual with parameter values $x_1 = 10.1$, $x_2 = 5.1$, for example, is being re-encoded in $R(X^{t1}_C)$ with 6 bits for each parameter; total length of a string in the child population being 12. Thus, the search space of the child population is 1/16 (= $2^{12}/2^{16}$)th of the original search space.

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



Phenotypic search space
$(x_{1,\min}, x_{2,\max})$
$(x_{1,\max}, x_{2,\max})$
$(x^{t1}_{1,c}-s_1/2, x^{t1}_{2,c}+s_2/2)$
$(x^{t1}_{1,c}+s_1/2, x^{t1}_{2,c}+s_2/2)$
$R(X^{t1}_C)$
s_2
s_1
$(x^{t1}_{1,c}-s_1/2, x^{t1}_{2,c}-s_2/2)$
$(x^{t1}_{1,c}+s_1/2, x^{t1}_{2,c}-s_2/2)$
$X^{t1}_C=(x^{t1}_{1,c}, x^{t1}_{2,c})$: The best solution up to $t1$
$R(X^{t1}_C)$: A neighborhood hypercube around X^{t1}_C
$(x_{1,\min}, x_{2,\min})$
$(x_{1,\max}, x_{2,\min})$
Forking
Child Population $CP^{t1}(t')$
Parent Population $PP^{t1}(t)$

Figure 9. Phenotypic population forking

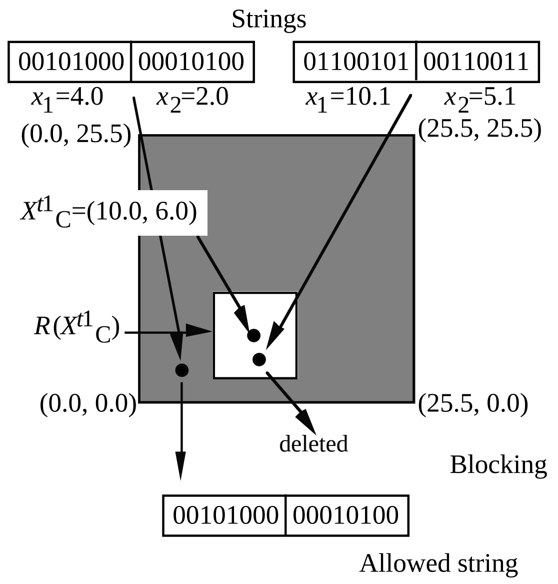


Figure 10. Blocking mode in the p-fGA

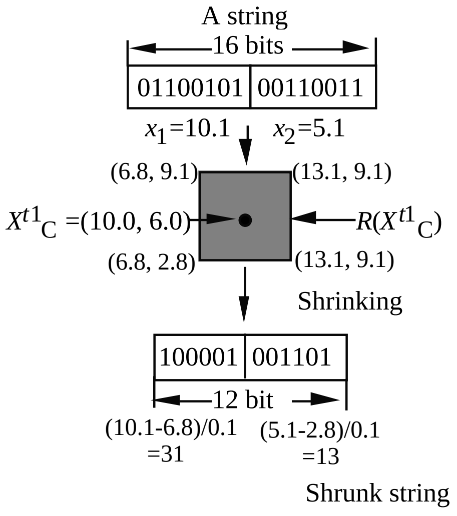


Figure 11. Shrinking mode in the p-fGA

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 n-fGA (non-forking GA: population forking is not applied), and the GENESIS (Grefenstette , Davis & Cerys, 1991) with an elite option. The performance was tested on the following four test functions.

i) De Jong's test function: f_1. This is a well known unimodal simple function (De Jong 1975) defined as

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

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 10×3 = 30 bit.

ii) 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. The length of a string is 3×10 = 30 bit.

iii) FMS (Frequency Modulation Sounds) parameter identification problem: f_{fms} (Tsutsui & Fujimoto, 1993) . Here the problem is to specify 6 parameters (a_1, w_1, a_2, w_2, a_3, w_3) of the FM sound 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:

$$y_0(t) = 1.0\times\sin(5.0t\theta - 1.5\times\sin(4.8t\theta + 2.0\times\sin(4.9t\theta))). \tag{9}$$

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 8×6 = 48 bit.

iv) Modified Griewank Function: f_{Griewank} (Torn & Zilmskas, 1989). The function is defined as follows:

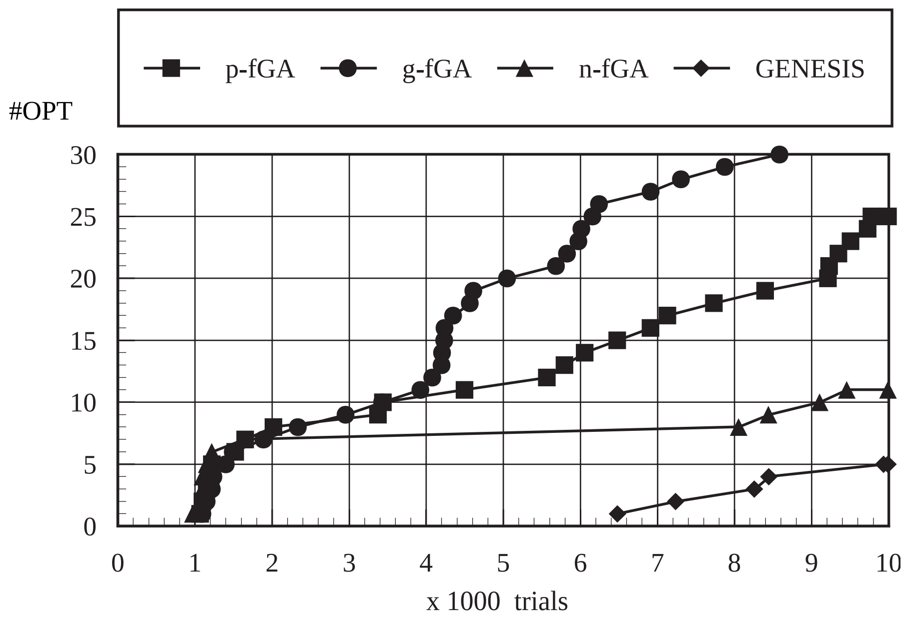
$$f_{\text{Griewank}} = \sum_{i=1}^{5} x_i^2 / 4000 - \prod_{i=1}^{5} \cos(x_i / \sqrt{i}) + 1 . \tag{10}$$

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 $10 \times 5 = 50$ bit.

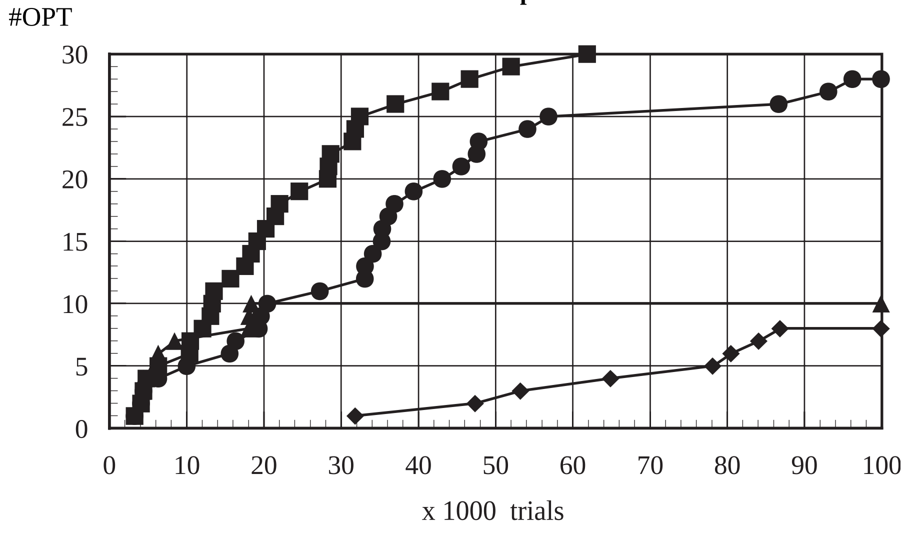
Maximum number of trials were set to 3,000, 10,000, 100,000, 140,000 for f_1, $f_{\text{deceptive}}$, f_{fms} and f_{Griewank}, respectively. 30 simulations were made for each experiment. Searching continued until the global optimum was found or the maximum number of trials was reached. A population size $N = 50$, Hamming power $\alpha = 0.05$, $P_{hm} = 0.1$ 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 the GENESIS; the mutation rate was tuned. Two point crossover operator was applied. We evaluated the models by measuring their #OPT (number of runs in which the algorithm succeeded in finding the global optimum) and MNT (mean number of trials to find the global optimum in those runs where it did find the optimum). Figure 12 shows the #OPT for restricted number of trials and Table 1 summarize the results after maximum number of trials.

For the function f_1, #OPT of the g-fAG, the p-fGA and the n-fGA were all 30. Also the MNT of the g-fGA and the p-fGA were almost the same as that of the n-fGA. Thus, the performance of all these algorithms were similar for the function f_1. For the function $f_{\text{deceptive}}$, the g-fGA performed better (#OPT = 30, MNT = 4,283.6) compared to the p-fGA (#OPT = 25, MNT = 5,332.8). The n-fGA (#OPT = 11, MNT = 3,426.7) and the GENESIS (#OPT = 5, MNT = 8,080.6) showed very poor performance. The performance of the g-fGA (#OPT = 28, MNT = 35,834.9) and the 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 the GENESIS performed bad. Similar results were produced by the g-fGA (#OPT = 30, MNT = 57,691.9) and the p-fGA (#OPT = 30, MNT = 43,599.4) for the function f_{Griewank}; and results of the n-fGA and the GENESIS were very poor. An overall analysis shows that the g-fGA and the p-fGA outperformed both the n-fGA and the GENESIS.

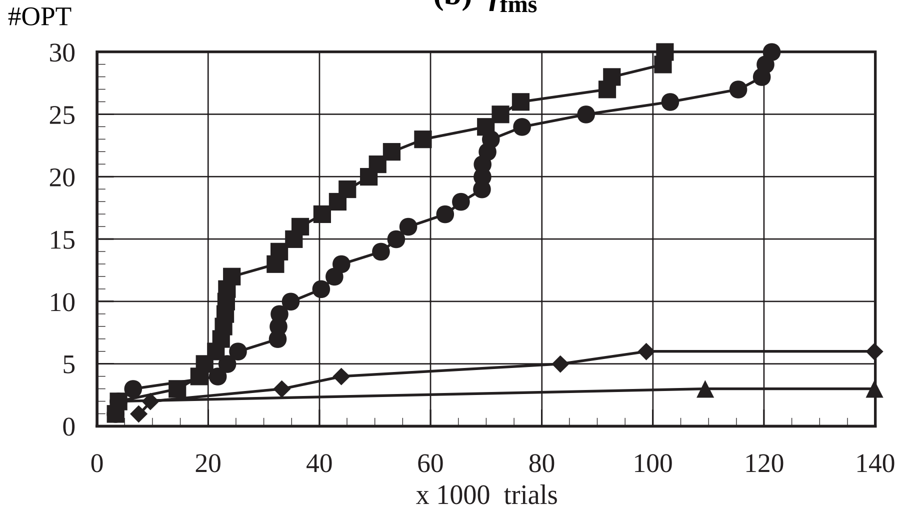
Performance of the g-fGA and the p-fGA depended on the type of the problem under consideration.



(a) $f_{\textbf{deceptive}}$



(b) $f_{\textbf{fms}}$



(c) $f_{\textbf{Griewank}}$

Figure 12. #OPT for restricted number of trials

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 bits used for each parameter in the child population

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

For the function $f_{\text{deceptive}}$, the optimum solution was always detected in the parent population by both the g-fGA and the 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 7 times in the child populations. Similar trend was seen for the function f_{Griewank} (g-fGA detected the optimum solution 8 times in the child populations and the p-fGA did the same 22 times). Thus for more complex functions (f_{fms} & f_{Griewank}), we see 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. This is also evident from the Figures 12 (b) & (c).

5. Two Other Utilities of the Phenotypic Forking GAs

We have noticed 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 scheme* and *niche-formation feature*.

5.1 Variable Resolution p-fGA

The p-fGA described in Subsection 3.2 uses the same resolution Δx_i for the parent and the child populations. Hereafter, we call this p-fGA as 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 as *variable resolution p-fGA (vp-fGA).* Thus the vp-fGA provides more flexibility to define 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 one 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 (please see Figure 13):

(a) parent population is searched with a lower resolution and detects the near optimal solution fast.

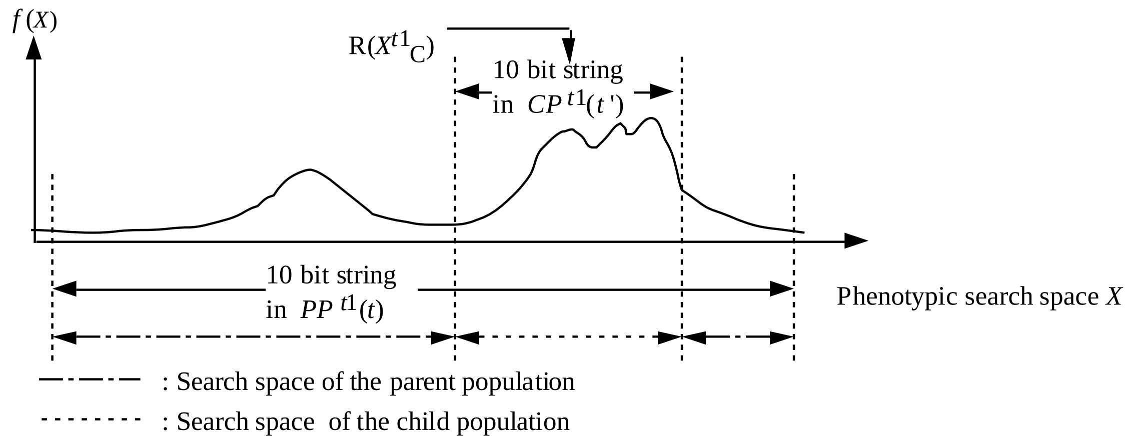

f (X)
R(X^t1_C)
10 bit string in CP^t1(t')
10 bit string in PP^t1(t)
Phenotypic search space X
: Search space of the parent population
: Search space of the child population

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

(b) 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.

In this context we mention that the variable resolution searching scheme is similar to the *dynamic parameter encoding* (DPE) technique (Schraudolph and 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 following two test functions f_{ripple} and $f_{\text{non-ripple}}$.

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

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

where, each x_i is in the range $0.0 \le x_i \le 100.0$, $i = 1, 2, .., 5$. The function f_{ripple} has many main peaks of different sizes surrounded by high frequency of small peaks. The function $f_{\text{non-ripple}}$ does not have a high frequency of small peaks. Both of these functions have the maximum value at $x_1 = x_2 =, .., x_5 = 0.1$ with functional value 5.5. We choose these functions because they require very high resolution to detect the actual optima. Let us consider 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

$x_1, x_2, .., x_5$ of the best individual are within the range [(0.1 - 0.0001) , (0.1 + 0.0001)].

The following experimental conditions are commonly used. 30 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 $K_R \in [0.5, 0.7]$, $K_H = \{60, 100\}$, $K_P \in [1, 3]$, $P_{nm} \in [0.006, 0.02]$, $\mathrm{BS}_{\mathrm{ratio}} \in [1:1, 3:1]$ and a total of 48 combinations are tried so that #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 bits 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 bits & 11 bits for its representation in the parent and the child populations, respectively. Thus, the resolution Δx_i of the parent and the child populations is 0.0000953 (= 100.0/ (2^{20}-1)), and $s_i = 0.195091$ (= 0.0000953$\times$(2^{11}-1)).

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

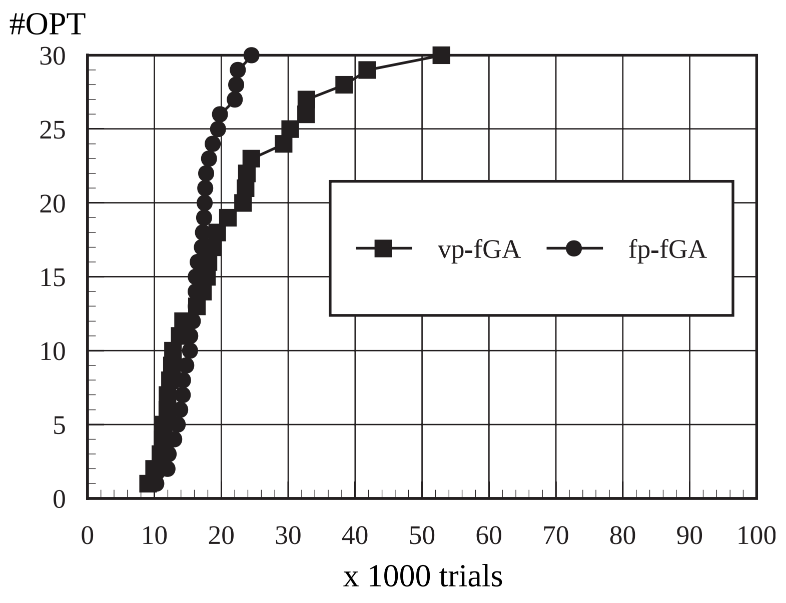
Thus, it is evident that the vp-fGA has a fairly good capability of finding the global optimum with high resolution. It may be mentioned here that with this feature of the vp-fGA, we can make compensation for the lack of local search capability of genetic algorithms.

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

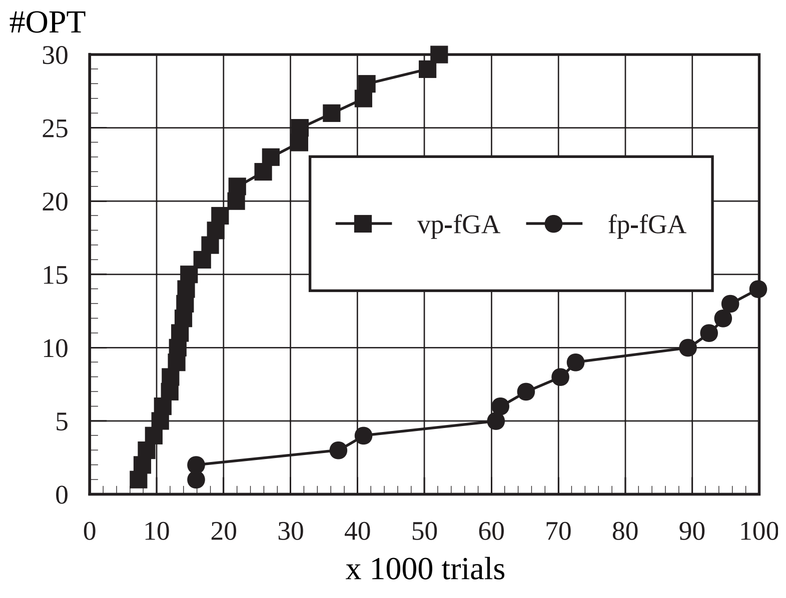
GA		fp-fGA	vp-fGA
String length	Parent Population	100(=20*5) bits	60(=12*5) bits
	Child Population	55 (=11*5) bits	55(=11*5) bits
Size of neighborhood hypercube s_i		0.1950791	0.15
Resolution Δx_i	Parent Population	0.0000953	0.02442
	Child Population	0.0000953	0.0000723
Other parameters		K_R=0.7, K_H=100, K_P=2, P_{nm}=0.02, BS_{ratio} =2:1	
$f_{\text{non-ripple}}$	**#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.



(a) $\boldsymbol{f}_{\textbf{non-ripple}}$



(b) $\boldsymbol{f}_{\textbf{ripple}}$

Figure 14. #OPT for restricted number of trials

5.2 The Niche-formation Feature of the p-fGA

We take a simple multi-modal function to show that the p-fGA has the niche-forming capability, defined as

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

This function was used by Deb & Goldberg (1989) to test their *sharing scheme* and it has five peaks of different sizes. We take the fp-fGA, and the parameters for its simulation are as follows: maximum number of trials for both the parent and the child populations was 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 $P_{hm} = 0.06$, maximum number of child populations $K_P = 5$, and the forking condition constant $K_R = 0.7$ & $K_H = 3$. 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, $(1.0/(2^{30} - 1))\times(2^{27} - 1) \approx 0.125$. This value is close to $\sigma_{\text{share}}\times 2$, where, σ_{share} is a sharing parameter defined by Deb & Goldberg (1989). Figure 15 shows a typical process of the niche-formation. With five child populations, all peaks are fairly covered with solutions. We did 30 runs for this problem. All of these experiments showed success in niche-formation, although the sequences of covering the peaks were different for different runs. Similar results were obtained with the vp-fGA also. Please note that parameters were chosen so as to ensure that a number of forkings occur.

6. Conclusions

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

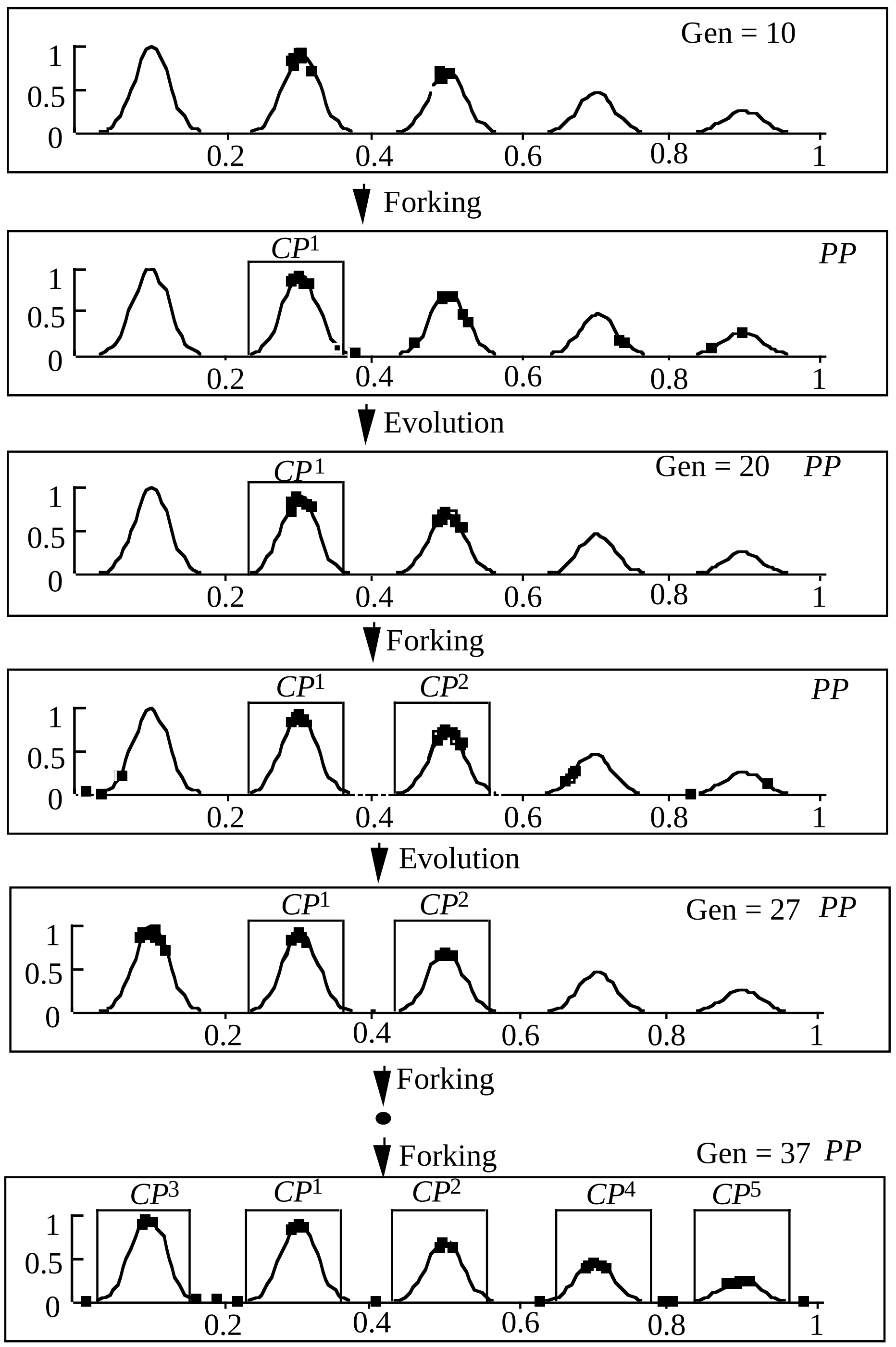


Figure 15. An example of niche-formation by the p-fGA

the phenotypic feature space.

The empirical results on some complex function optimization problems showed that both the g-fGA and the p-fGA perform fairly well over the conventional GAs. Although performance of the g-fGA and the p-fGA depends on the type of the problem, for more complex functions we see that the p-fGA maintains a balance in exploring the parent population and exploiting the child populations; and shows better performance than the g-fGA. The fact that use of phenotypic search space information outperforms the use of genotypic search space information corroborates the earlier finding of Deb & Goldberg (1989) on *sharing scheme* where, phenotypic sharing maintained better subpopulation than the genotypic sharing. Two additional utilities of the p-fGA are also studied briefly. Although the p-fGA is mainly better than the g-fGA, for problems where genotypic feature directly maps to fitness value the g-fGA will be useful.

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 to discard some of the child populations. Evaluating the effectiveness of the fGAs on real life problems, comparing them with other multi-population based schemes, extending them for permutation problems & 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. Further, analytical analysis may establish some relation of the p-fGA with the global random searching technique (Peck & Dhawan, 1995).

Acknowledgments

We like to 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 for Scientific Research on Priority Areas number 264-08233105.

References

Beasley, D., Bull, D. R. & Martin, R. R. (1993). A sequential niche technique for multimodal function optimization, *Evolutionary Computation*, *1*(2), pp. 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 Internationa*l *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. *Ph. D. dissertation, Dept. 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., Korb, B. & Deb, K. (1989). Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 3, pp. 493-530.

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

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.

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

Mathias, K & Whitely, 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 Service Center.

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

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

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

Torn, A. & Zilmskas, A. (1989). In G. Goos and 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 Algorithm*s (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 Service Center.

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. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp. 221-241). San Mateo, CA: Morgan Kaufmann.

ﾅD