ELSEVIER Journal of Information Sciences 109 (1998) 119–133

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

Search space division in GAs using phenotypic properties

Shigeyoshi Tsutsui [a,*], Ashish Ghosh [b,1]

[a] *Department of Management and Information Science, Hannan University, 5-4-3 Amamihigashi, Matsubara, Osaka 580, Japan*

[b] *Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700035, India*

Received 17 January 1997; received in revised form 29 September 1997; accepted 12 November 1997

Abstract

In this article, we study a new type of *forking GA (fGA)*, the *phenotypic forking* GA (p-fGA). The fGA divides the whole search space into subspaces depending on the convergence status of the population and the solutions obtained so far; and is intended to deal with multimodal problems which are difficult to solve using conventional GA. 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. The p-fGA divides the search space using phenotypic properties only, and defines a search subspace (to be exploited by a child population) by a *neighborhood hypercube* around the current best individual in the phenotypic feature space. Empirical results on complex function optimization problems show that the p-fGA performs fairly well compared to a conventional GA. Two other variants of the p-fGA, *the moving window p-fGA* (to accelerate the speed of convergence in the child populations) and the *variable resolution p-fGA* (to solve multimodal problems with high precision) are also studied in this article.

Keywords: Search space division; Phenotypic properties; multi-modal problems; Forking GA

* Corresponding author. Tel.: +81 723 32 1224; fax: +81 723 36 2633; e-mail: tsutsui@hannan-u.ac.jp.

[1] Tel.: +91 33 577 8085 (ext 3110); fax: +91 33 556 6680; e-mail: ash@isical.ernet.in.

0020-0255/98/$19.00
PII: S0020-0255(98)00013-9

1. Introduction

There are many GA-hard problems (such as multimodal problems and deceptive functions [5,13]) that are difficult to be solved by the conventional GAs. Various kinds of modified GAs, such as CHC [3], messy GA (mGA) [6], delta coding [8], niche methods [1,2] and fGA [11,12] aimed to solve these problems are proposed in the literature.

The *forking GA (fGA)* [11,12] divides the search space into subspaces depending on the status of convergence of the present population and the solutions obtained so far; and uses a multi-population scheme with one parent population for *exploration* and one or more child populations for *exploitation*, generated by *population forking*. Depending on the type of the search space to be divided, different types of fGAs can be considered. In the present work we study one such type, the *phenotypic fGA (p-fGA)* which uses phenotypic properties for space division where each subspace is defined by a *neighborhood hypercube* around the current best individual in the phenotypic parameter space.

Empirical results on multimodal function optimization problems showed that the p-fGA performs fairly well over a conventional GA. We also studied two variants of the p-fGA. One of them is the *moving window p-fGA (mp-fGA)* which accelerates the speed of convergence in the child populations. The other is the *variable resolution searching scheme (vp-fGA)* aimed to solve multimodal problems with high precision. Results are found to be encouraging for them also.

2. Basic evolutionary model

Although the basic principle of fGAs does not depend on any particular 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 and normal mutation* or *high mutation* followed by *population elitist selection*; and is shown in 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} (= Hamming distance $(S_i, S_j)/L$; L: length of an individual, S_i: an individual) between a chromosome pair (S_i, S_j) in $P(t-1)$. Crossover of this pair is done with probability $(H_{ij})^{\alpha}$, where α $(0 < \alpha \leqslant 1)$ is called the *crossover Hamming power*; normal mutation with rate P_{nm} is applied after crossover. Offspring thus generated is stored in offspring population $C(t-1)$. When crossover is not performed, a high mutation with rate $P_{\mathrm{hm}}(P_{\mathrm{hm}} \gg P_{\mathrm{nm}})$ is applied to the individual with lower functional value 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 $C(t-1)$. This selection

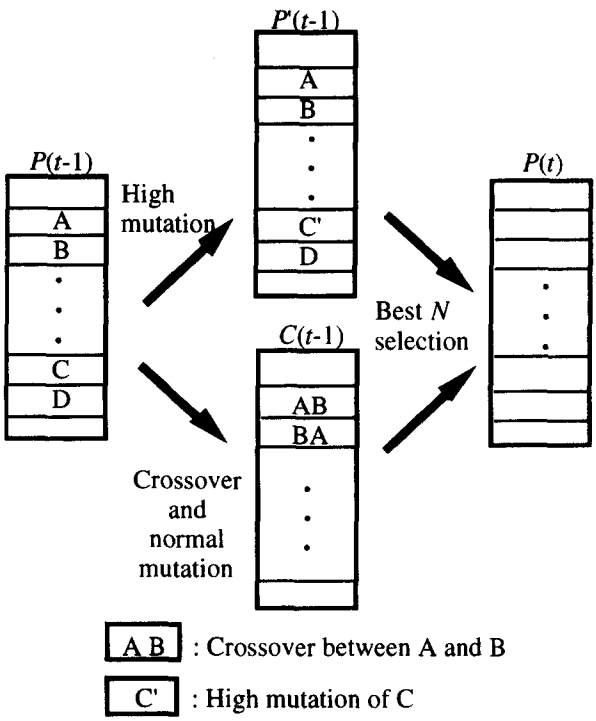

Fig. 1. Basis model of evolution.

method is called *population elitist selection* [3], 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; and consequently a high mutation is performed. Thus an appropriate amount of diversity can be maintained in the population by a proper choice of α. In this context, we mention that attempts are also made [4] to maintain diversity in the population.

3. Search space division by population forking

In this section we describe the basic idea of population forking for search space division.

3.1. Population forking

During the process of evolution if the population is found to be converged (measured by different parameters), the process may be forked to allow searching concurrently in two different subpopulations. Thus the entire search space

is divided into subspaces depending on the status of convergence of the present population and the solutions obtained so far; and search is continued independently in these subspaces. We call this method *population forking*. The population is forked into a *parent population* $PP^{t1}(t)$ and a *child population* $CP^{t1}(t')$ covering the subspaces as shown in Fig. 2. The parent population and the child populations are evolved in time sharing mode.

If the conditions for forking is satisfied again during the evolution of the parent population, the second child population is formed. A maximum of K_P ($\geqslant 1$) child populations are allowed. Sharing of computational time by the parent and child populations is defined by the PC_{ratio} on the generation counter. For example, the $PC_{ratio} = p : q$ 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 child populations. But when an individual with the new best solution is found in a child population, it is copied into 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.

3.2. Search space division using phenotypic properties

In the present work a subspace (to be searched by a child population) is defined by a neighborhood hypercube in the phenotypic feature space around the current best solution as described in the following paragraphs.

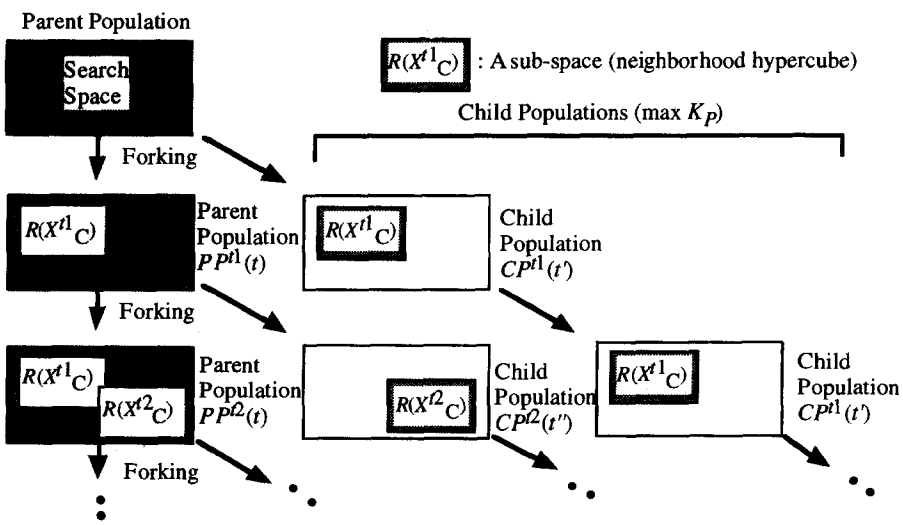

Fig. 2. Population forking.

Let the phenotypic parameter of a problem be $\mathbf{X} = (x_1, x_2, \ldots, x_n)$. 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 $X_C^t = (x_{1,c}^t, x_{2,c}^t, \ldots, x_{n,c}^t)$. Thus, the neighborhood hypercube $R(X_C^t)$ around X_C^t be defined as $R(X_C^t) = \{x_1, x_2, \ldots, x_n \mid (x_{i,c}^t - s_i/2) \leqslant x_i \leqslant (x_{i,c}^t + s_i/2), i = 1, 2, \ldots, n\}$, where $\mathbf{S} = (s_1, s_2, \ldots, s_n)$ defines the size of the neighborhood hypercube $R(X_C^t)$ and s_i (> 0) is a user specified constant.

A population is allowed to fork if the following conditions are satisfied simultaneously.

1. the current best individual has not been updated by a new individual for a specified number $(K_H \geqslant 1)$ of generations, and
2. the number of the individuals located inside the neighborhood hypercube $R(X_C^t)$ is more than a specified number $N \times K_R$ $(0 < K_R \leqslant 1.0)$.

If the conditions of forking are satisfied, we allow the initial population to fork into a parent population $\mathrm{PP}^{t1}(t)$ which evolves outside $R(X_C^{t1})$, and a child population $CP^{t1}(t')$ which evolves inside $R(X_C^{t1})$ (refer Fig. 2). Since we use phenotypic properties for measuring convergence and phenotypic features for space division, we refer this technique as *phenotypic forking GA (p-fGA)*.

3.3. Exploratiom and exploitation of subspaces

After a population forking has occurred, individuals which are located inside $R(X_C^{t1})$, except the best one, will be deleted from the parent population $\mathrm{PP}^{t1}(t)$ and we call this the *blocking mode*. Individuals are randomly regenerated to keep the parent population size fixed; and thus the diversity of $\mathrm{PP}^{t1}(t)$ is recovered and it starts *exploring* other sub-spaces. Fig. 3 shows an example of the blocking mode. In this figure, there are two phenotypic parameters, x_1 and x_2 in the range $0.0 \leqslant x_1, x_2 \leqslant 25.5$ and coded by 8 bits. We assume $X_C^{t1} = (10.0, 6.0)$. Let the *resolution* of parameter x_i be represented by Δx_i. Then $\Delta x_1, \Delta x_2$ are both 0.1 $(= (25.5 - 0.0)/(2^8 - 1))$.

The *neighborhood hypercube size* $\boldsymbol{S}$ can be determined from the number of bits used to represent each of the parameters in the child population and the resolution. If six bits are used to represent both x_1 and x_2 in the child population, then $\mathbf{S}$ becomes $((2^6 - 1) \times 0.1, (2^6 - 1) \times 0.1) = (6.3, 6.3)$; and thus $(10.0 - 3.1) \leqslant x_1 \leqslant (10.0 + 3.2)$ & $(6.0 - 3.1) \leqslant x_2 \leqslant (6.0 + 3.2)$, as shown in Fig. 4. An individual with parameter values $x_1 = 10.1, x_2 = 5.1$, for example, is being re-encoded in $R(X_C^{t1})$ 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. As the string length of the chromosomes is reduced, we call this *shrinking mode*. The child populations thus *exploit* the small search spaces to detect the actual optimum.

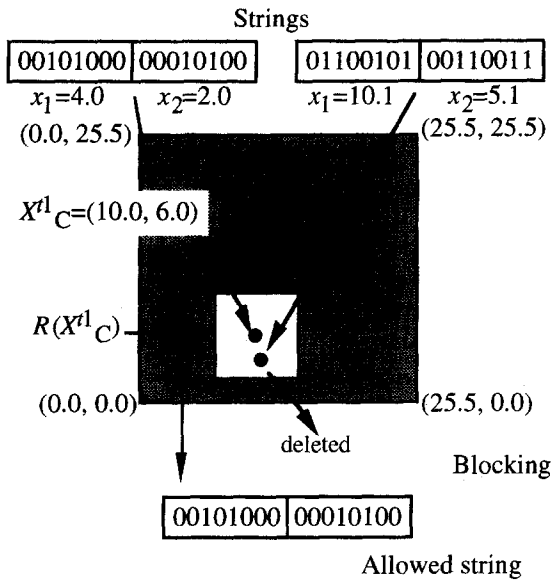

Fig. 3. Blocking mode in the p-fGA.

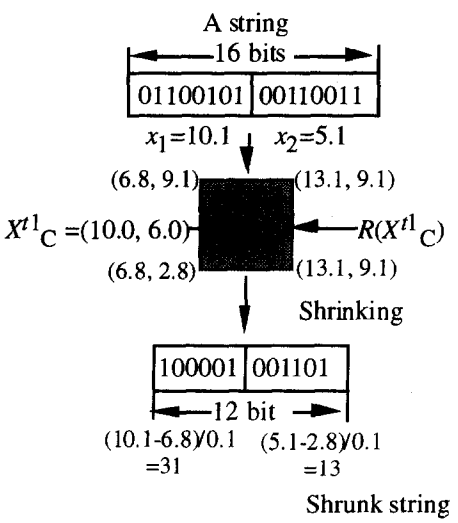

Fig. 4. Shrinking mode in the p-fGA.

4. Empirical results

In this section, experimental results are analyzed to evaluate the p-fGA. Two more GAs are tested. They are the n-fGA (non-forking GA: population

forking is not applied), and the GENESIS [7] with elitism. The performance was tested on the following four multimodal functions. Each of these functions has a large number of local optima and a single global optimum and has various amounts of complexity.

(i) *Sine envelope sine wave:* f_6. The function is defined [14] as follows.

$$f_6 = 0.5 + \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{(1.0 + 0.001(x_1^2 + x_2^2))^2}. \tag{1}$$

Each parameter x_i is represented by a 22-bit Gray code in the range -100–100. The length of a string is thus $22 \times 2 = 44$ bits. The minimum value of this function is 0.0.

(ii) *Stretched V sine wave:* f_7. The function is defined [14] as follows.

$$f_7 = (x_1^2 + x_2^2)^{0.25}(\sin^2(50(x_1^2 + x_2^2)^{1.0} + 1.0)). \tag{2}$$

Here also each parameter x_i is represented by a 22-bit Gray code in the range -100–100. Thus the length of a string is $22 \times 2 = 44$ bits. The minimum value of this function is also 0.0.

(iii) *FMS (Frequency Modulation Sounds) parameter identification problem:* f_{fms} [12]. 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

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_3 \sin(w_3 t\theta))), \tag{3}$$

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

$$f_{\text{fms}} = \sum_{t=0}^{100} (y(t) - y_0(t))^2, \tag{4}$$

where the model data are given by the following equation.

$$y_0(t) = 1.0 \times \sin(5.0t\theta - 1.5 \times \sin(4.8t\theta + 2.0 \times \sin(4.9t\theta))). \tag{5}$$

Each parameter is represented by an 8-bit Gray code in the range -6.4–6.35. Hence the total length of a string is $8 \times 6 = 48$ bits.

(iv) *Modified Griewank Function:* f_{Griewank} [10]. The function is defined as follows:

$$f_{\text{Griewank}} = \sum_{i=1}^{5} x_i^2/4000 - \prod_{i=1}^{5} \cos(x_i/\sqrt{i}) + 1. \tag{6}$$

Here each parameter x_i is represented by a 10-bit Gray code in the range -51.2–51.1. The total length of a string is $10 \times 5 = 50$ bits. The minimum value of this function is 0.0.

Maximum number of trials (function evaluations) were set to 100,000, 100,1000, 140,000 and 200,000 for f_6, f_7, 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 $\alpha = 0.05$, a normal mutation rate $P_{nm} = 0.02$, a high mutation rate $P_{hm} = 0.2, K_p = 3$, $PC_{ratio} = 3{:}1$ and $K_H = 60$ were commonly used for all the experiments. The number of bits used for each parameter in the child population was tuned, and were 17, 10, 5 and 7 for f_6, f_7, f_{fms} and $f_{Griewank}$, respectively. Except for the mutation rate, we used the default parameter values for experiments with GENESIS; where a mutation rate of 0.02 was used. The 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) which indicates the *success rate*; and MNT (mean number of trials to find the global optimum in those runs where it did find the optimum) reflecting the *convergence rate* for detecting the global optima. Fig. 5 shows the OPT for restricted number of trials and Table 1 summarizes the results after maximum number of trials.

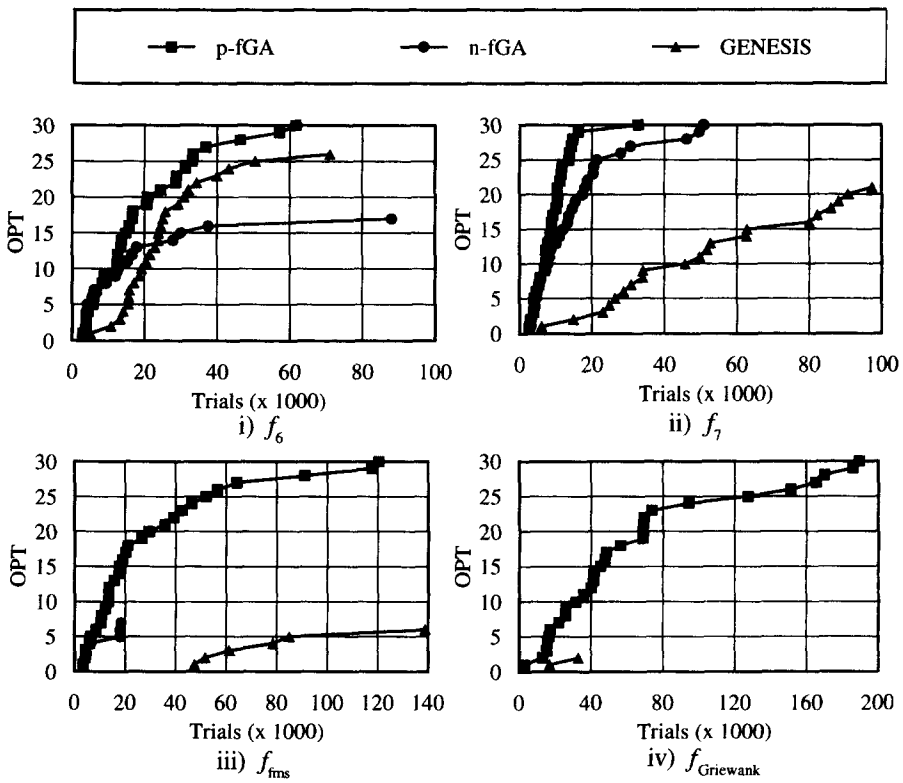

Fig. 5. OPT for restricted number of trials.

Table 1
Performance of the p-fGA

GA	Function	f_6	f_7	f_{fms}	f_{Griewank}
p-fGA	OPT	30	30	30	30
	MNT	20 097.7	9 532.4	31 338.2	65 864.4
n-fGA	OPT	17	30	7	1
	MNT	17 742.6	15 961.7	10 460.6	2910.0
GENESIS	OPT	26	21	6	2
	MNT	25 549.6	51 121.9	77 134.3	24 849.5

For these four functions, the p-fGA showed better performance than the n-fGA and GENESIS. The p-fGA found the global optimum solution 30 times (OPT = 30) for all of the functions. On the function f_6, GENESIS (OPT = 26, MNT = 25,549.6) showed similar performance to the p-fGA (OPT = 30, MNT = 20,097.7), but the n-fGA (OPT = 17, MNT = 17,742.6) showed poorer performance than the p-fGA. On the other hand, for the function f_7, the n-fGA (OPT = 30, MNT = 15,961.7) showed similar performance to the p-fGA (OPT = 30, MNT = 9532.4), but GENESIS (OPT = 21, MNT = 51,121.9) did not do well. On the functions f_{fms} and f_{Griewank}, the p-fGA showed good performance (OPT = 30, MNT = 31,338.2 and OPT = 30, MNT = 65,864.4, respectively). The n-fGA detected the global optimum only 7 times for f_{fms}, and once for f_{Griewank}; and GENESIS did the same 6 and 2 times, respectively. Thus, we can see that the p-fGA has very stable higher performance on these four multimodal functions than the other GAs; either in terms of detecting the optimal solution (Table 1), or the convergence rate of doing the same (Fig. 5).

5. Variants of the p-fGA

We have noticed from the results in Section 4 that the p-fGA improves the performance by detecting the optimal solution more frequently and quickly. In this section we consider two variations of the p-fGA; the *moving window pf-GA (mp-fGA)* and the *variable resolution searching scheme* to improve the speed of convergence, and to produce high precision search.

5.1. Moving window p-fGA (mp-fGA)

In the original p-fGA (Section 3), the neighborhood hypercube is defined around the best individual obtained at the time of forking. Hence, the search subspaces remain fixed during the remaining portion of the algorithm. Thus, detection of the actual position of the optimum becomes largely dependent on the size of the neighborhood hypercube. If the neighborhood hypercube

is small, we may miss the actual location of the optimum or the optimum itself. In other words, the optimum may not be detected in the child population(s). On the contrary, if the size of the neighborhood hypercube is large, we may get the actual optimum; but searching becomes very expensive. Thus, the choice of the neighborhood hypercube can become a bottleneck for the p-fGA. In this study, we shift the position of the neighborhood hypercube with time. The center of the hypercube is updated every generation to be the current best solution; thereby dynamically varying the search space for the parent and the child populations. This gives us more scope to detect the actual optimum in the child populations; thereby increasing the chance of detecting the actual optimum in fewer number of trials. We call this technique the *moving window p-fGA (mp-fGA)*.

The performance of the mp-fGA was tested on the same set of test functions as in Section 4. Here also we used OPT and MNT as performance measure. Fig. 6 shows the OPT for restricted number of trials. Table 2 summarizes the results. The mp-fGA performed better than the p-fGA for all four test functions. The mp-fGA also found the global optimum 30 times for all of the four functions. Furthermore, the MNT of the mp-fGA for each function is less than

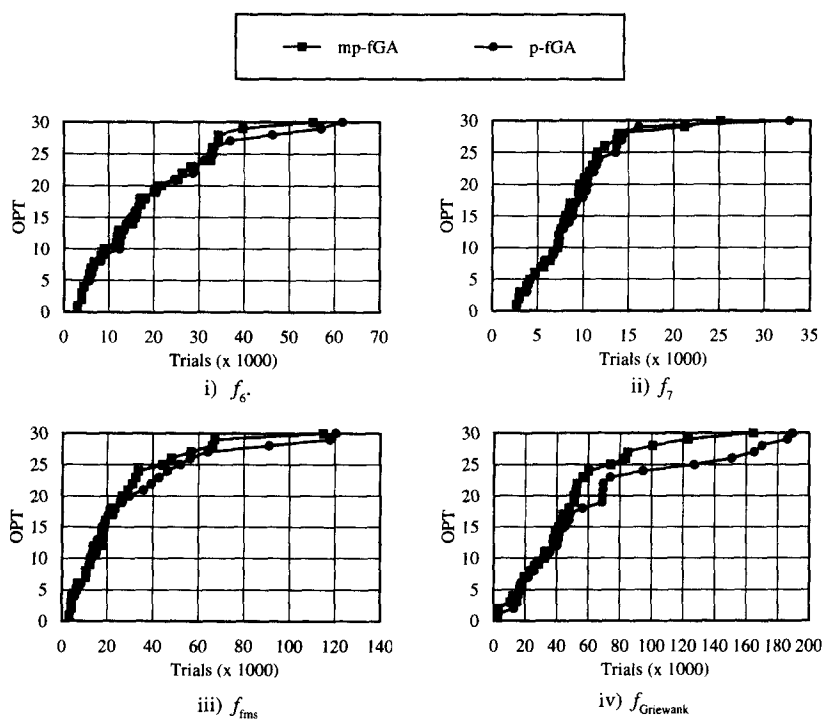

Fig. 6. OPT for restricted number of trials.

Table 2
The mp-fGA vs. the p-fGA

GA	Function	f_6	f_7	f_{fms}	f_{Griewank}
mp-fGA	OPT	30	30	30	30
	MNT	18 590.9	9078.9	26 747.3	48 145.1
	OPT/C [a]	20	21	8	21
p-fGA	OPT	30	30	30	30
	MNT	20 097.7	9532.4	31 338.2	65 864.4
	OPT/C [a]	18	19	6	20

[a] Number of runs in which the optimal solution was found in one of the child populations.

that of the p-fGA. Thus, the mp-fGA enhances performance by reducing the MNT (by approximately 5–27%) and thereby accelerates the speed of convergence.

The performance improvement of the mp-fGA can be explained from the OPT/C (number of runs which found the global optimum in one of the child populations) in Table 2. For the function f_6, OPT/C of the p-fGA = 18, OPT/C of the mp-fGA = 20. Similar results were also found for f_7 (OPT/C = 19 for the p-fGA; OPT/C = 21 for the mp-fGA), f_{fms} (OPT/C = 6 for the p-fGA; OPT/C = 8 for the mp-fGA) and f_{Griewnak} (OPT/C = 20 for the p-fGA; OPT/C = 21 for the mp-fGA). Thus we see, the mp-fGA found the global optimum more number of times in the child populations than that of the p-fGA; and thus required fewer trials to detect the global optimum. Results obtained with other neighborhood hypercube sizes also corroborated the earlier finding.

5.2. *Variable resolution p-fGA*

The p-fGA described in Section 3 uses the same resolution Δx_i for the parent and the child populations. In this section, 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 a *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 (Fig. 4) with the fp-fGA. 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 $\boldsymbol{S}$ and a given number of bits to represent the members of a child population. Thus, we can take any value for $\boldsymbol{S}$ 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.

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

In this context we mention that the variable resolution searching scheme is similar to the *dynamic parameter encoding* (DPE) technique [9]; however 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 $f_{\text{non-ripple}}$:

$$f_{\text{ripple}} = \sum_{i=1}^{5} e^{-2\ln 2\left(\frac{x_i-0.1}{0.8}\right)^2} \left(\sin^6(5\pi x_i) + 0.1 \times \cos^2(500\pi x_i)\right), \tag{7}$$

$$f_{\text{non-ripple}} = \sum_{i=1}^{5} e^{-2\ln 2\left(\frac{x_i-0.1}{0.8}\right)^2} \sin^6(5\pi x_i), \tag{8}$$

where, each x_i is in the range $0.0 \leqslant x_i \leqslant 100.0, i = 1, 2, \ldots, 5$. The function f_{ripple} has many main peaks of different sizes surrounded by a high frequency of small peaks; the function $f_{\text{non-ripple}}$ does not have a high frequency of small peaks. Both of these functions have their maximum value at $x_1 = x_2 = , \ldots, x_5 = 0.1$ with functional value 5.5. We choose these functions because they require a 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 $x_1, x_2, \ldots, x_5$ of the best individual are within the range $[(0.1 - 0.0001), (0.1 + 0.0001)]$.

The following experimental conditions are commonly used. Thirty runs are performed, where 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 a two point crossover are used. Other parameters are tuned so that the OPT of the fp-fGA for function f_{ripple} is maximized. The size of the neighborhood hypercube ($\boldsymbol{S}$) was set close to the diameter ($= 0.15$) of the main peaks.

In the vp-fGA, we used $s_i = 0.15$ for all i. To represent each parameter x_i, 12 and 11 bits were used in the parent and child populations, respectively. Thus, the resolution Δx_i of the parent and child populations were 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 the child populations, respectively. Thus, the resolution Δx_i of the parent and the child populations was 0.0000953 $(= 100.0/(2^{20} - 1))$, and $s_i = 0.195091$ $(= 0.0000953 \times (2^{11} - 1))$.

Table 3
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	$K_R = 0.7, K_H = 100, K_p = 2,$ $P_{nm} = 0.02$, $BS_{ratio} = 2{:}1$	
$f_{non\text{-}ripple}$		
OPT	30	30
OPT/P [a]	2	–
OPT/C [b]	28	30
MNT	16,845.7	20,916.0
f_{ripple}		
OPT	14	30
OPT/P [a]	4	–
OPT/C [b]	10	30
MNT	65,272.9	21,087.4

[a] Number of runs in which the optimal solution was found in the parent population.
[b] Number of runs in which the optimal solution was found in one of the child populations.

Simulation results are shown in Table 3. For function $f_{non\text{-}ripple}$ the results of the fp-fGA and vp-fGA are almost the same; the OPTs of the fp-fGA and the vp-fGA were both 30 (100%), the MNT of the fp-fGA and vp-fGA were 16,845.7 and 20,916.0, respectively. For the function f_{ripple}, the vp-fGA showed better performance; the OPT of the vp-fGA was 30 (100%), and that of the fp-fGA was 14 (47%) only; the MNTs of the fp-fGA and vp-fGA were 65,272.9 and 21,087.4, respectively. It can be mentioned here that the n-fGA and GENESIS could not find the global optimum in any of the 30 runs for these functions.

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.

6. Conclusions

In the present work we study the *phenotypic fGA* (p-fGA) which uses phenotypic properties for space division where each subspace for a child population

is defined by a *neighborhood hypercube* around the current best individual in the phenotypic parameter space. Empirical results on multimodal function optimiization problems showed that the p-fGA performs fairly well over the conventional GAs.

We also studied two other variants of the p-fGA. One of them is the *moving window p-fGA* (*mp-fGA*) which accelerates the speed of convergence in the child populations. Empirical results on complex function optimization problems showed that the new method found the global optimum in less (by approximately 5–27%) number of trials than the original p-fGA. The other is the *variable resolution searching scheme* (*vp-fGA*) to solve multimodal problems with high precision. The empirical results showed that the vp-fGA had a fairly good capability to finding the global optimum with high resolution.

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. Finally work remains in evaluating the effectiveness of the p-fGAs on real life problems, comparing them with other multi-population based schemes, extending them for permutation problems and other evolution schemes such as real coded GAs.

References

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

[2] K. Deb, D.E. Goldberg, An investigation of niche and species formation in genetic function optimization, in: Proceedings of Third International Conference on Genetic Algorithms, 1989, pp. 42–50.

[3] L.J. Eshelman, The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, in: Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 265–283.

[4] L.J. Eshelman, J.D. Schaffer, Preventing premature convergence in genetic algorithms by preventing incest, in: Proceedings of Fourth International Conference on Genetic Algorithms, 1991, pp. 115–122.

[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

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

[7] J.J. Grefenstette, L. Davis, D. Cerys, GENESIS and OOGA: Two GA Systems, TSP Publication, Melrose, MA, 1991.

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

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

[10] A. Torn, A. Zilmskas, Global optimization, in: Lecture Notes in Computer Science, Springer, Berlin, 1989, p. 186.

[11] S. Tsutsui, Y. Fujimoto, Forking genetic algorithm with blocking and shrinking modes, in: Proceedings of Fifth International Conference on Genetic Algorithms, 1993, pp. 206–213.
[12] S. Tsutsui, Y. Fujimoto, A. Ghosh, Forking GAs: GAs with search space division schemes, Evolutionary Computation 5 (1) (1997) 61–80.
[13] D. Whitley, Fundamental principles of deception in genetic search, in: Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1991, pp. 221–241.
[14] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, R. Das, A study of control parameters affecting online performance of genetic algorithms for function optimization, in: Proceedings of Third International Conference on Genetic Algorithms, 1989, pp. 51–60.