ICONIP-96

Phenotypic Forking GA with Moving Windows

Shigeyoshi Tsutsui[*1], Ashish Ghosh[*2] and Masato Takiguchi[*1]

*1 Department of Management and Information Science
Hannan University
5-4-33 Amamihigashi, Matsubara, Osaka 580 Japan
tsutsui@hannan-u.ac.jp
*2 Department of Industrial Engineering, College of Engineering
Osaka Prefecture University
1-1 Gakuen-cho, Sakai, Osaka 593 Japan

Abstract--- The *phenotypic forking GA (p-fGA)* which divides the whole search space into sub-spaces using the information of the convergence status of the population and the solutions obtained so far had already been developed. In that work, a *neighborhood hypercube* was defined around the best individual at the time of forking in the *phenotypic feature space* with the best solution as the center of that hypercube; and that sub-space was searched in *exploitation mode*. The rest of the search space was *explored*. In this investigation, we extend this concept by moving the center of the neighborhood hypercubes with generations. Here, the position of the center of the hypercube is updated every generation such that it becomes the current best solution; thereby *dynamically modifying the sub-spaces* for exploration and exploitation. This enhances the scope of tracing the optimum solution and gives more flexibility for choosing the size of the hypercube. Empirical results on complex function optimization problems show that the new method finds the global optimum in less number of trials than the original p-fGA.

1 Introduction

There are many GA-hard problems that are difficult to be solved by the conventional GAs such as multi-modal problems and deceptive functions [3], [10]. Many kinds of modified GAs, such as CHC [2], mGA [4], delta coding [6], niche methods [1] and fGA [8], [9] aimed to solve these problems are proposed in the literature.

The f*orking GA (fGA)* divides the search space into sub-spaces depending on the status of convergence of the present population and the solutions obtained so far; and uses 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*. Depending on the type of the search space to be divided two types of fGAs are proposed. One of them 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 then best individual in the phenotypic parameter space.

In the p-fGA, since the position of a neighborhood hypercube is fixed after it is created, efficiency of searching the optimal point in the neighborhood hypercube becomes lower when it is located near the edge of the hypercube or just outside of it. Thus the size of the hypercube plays a major role in detecting the actual position of the optima and time needed for the same. In the present work an attempt is made to solve these problems by updating the position of the center of the neighborhood hypercube every generation. The current best solution is set to be the center of the neighborhood hypercube; thereby *dynamically modifying the sub-spaces* for exploration and exploitation. This enhances the scope of tracing the optimum solution and gives more flexibility for choosing the size of the hypercube.

The empirical results on complex function optimization problems show that the new method finds the global optimum in less (by 20-30 %) number of trials than the original p-fGA.

2 Population Forking

During the process of evolution, if the population is converged to a smaller diversity, the process may be forked to

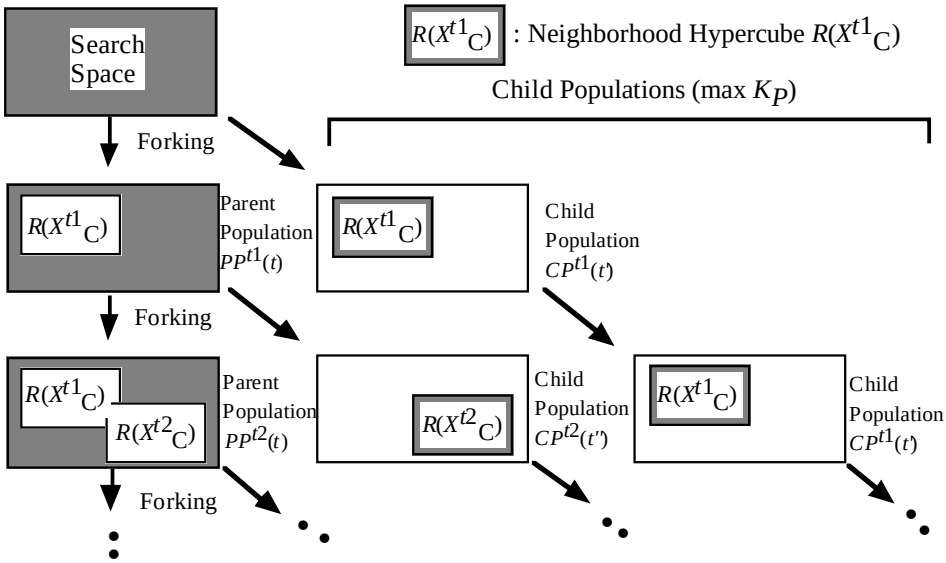


Fig. 1 Population forking

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*. The population is forked into a *parent population* $PP^{t1}(t)$ and a *child population* $CP^{t1}(t')$ covering the sub-spaces as shown in Fig. 1.

If the conditions for forking is satisfied again in the parent population, the second child population is formed. A maximum of 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 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 the child population; 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.

3 Phenotypic Population Forking

In the p-fGA, a subspace (child population) is defined by a neighborhood hypercube in the phenotypic search space around the current best solution as described bellow.

Let the phenotypic parameter of a problem be $\boldsymbol{X} = (x_1, x_2, ..., x_n)$. 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 $\boldsymbol{S} = (s_1, s_2, ..., s_n)$ 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>1) of generations, and

(ii) the number of the individuals located inside the neighborhood hypercube $R(X^t_C)$ is more than the specified number $N \times K_R$ ($0 < K_R \leq 1.0$).

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 $R(X^{t1}_C)$ (refer Fig. 1).

3.1 Exploration and Exploitation of Sub-spaces

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 $PP^{t1}(t)$ and we call this as *blocking mode*; 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. Fig. 2 shows an example of the blocking mode in the p-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 *resolution* 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 resolution. 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 & 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 (10.0-3.1) $\leq x_1 \leq$ (10.0+3.2) & (6.0-3.1) $\leq x_2 \leq$ (6.0+3.2) as shown in Fig. 3. Exploration in the parent population is then continued.

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. As the string length of the chromosomes is reduced, we call this shrinking mode. The child population is then exploited to detect the actual optimum.

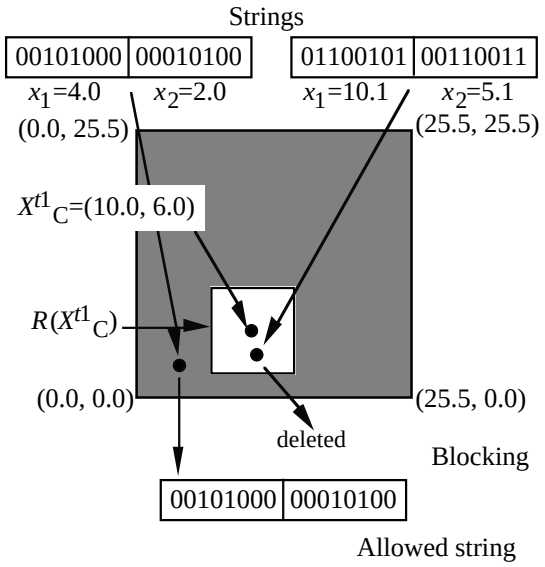


Fig. 2 Blocking mode

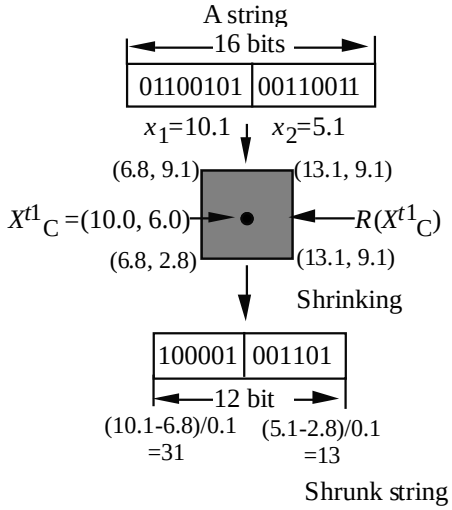


Fig. 3 Shrinking mode

3.2 Variable Resolution p-fGA

The p-fGA described above 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. This can only be attained by increasing

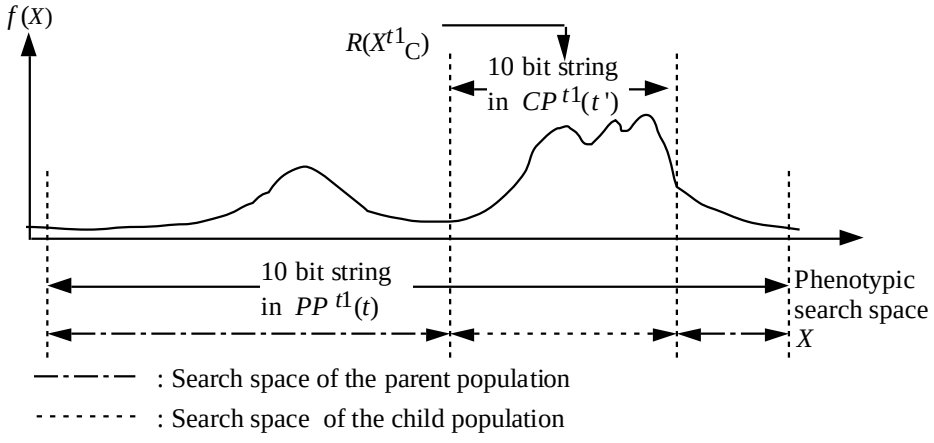


Fig. 4 Variable resolution searching in the vp-fGA

the number of bits to represent strings of the child population. However, if we increase one bit to represent x_1, for example, 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. 4):

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

(b) in the child populations searching is performed with a higher resolution, depending on the problem, resulting in the efficient detection of the global optimum or local optima, since searching proceeds in a smaller phenotypic search space.

4 Moving Window p-fGA (mp-fGA)

In the original p-fGA, the neighborhood hypercube is defined around the best individual obtained at the time of forking. Hence, the search sub-spaces remains fixed during the rest 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 populations. On the contrary, if the size of the neighborhood hypercube is more, we may get the actual optimum; but searching becomes very high. Thus, the choice of the neighborhood hypercube becomes a bottleneck of the p-fGA. In the present investigation, we move the neighborhood hypercube with time. The center of the hypercube is updated every generation to be the current best optimum; 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 less number of trials.

5 Empirical Results and Discussion

In this section, performance of the mp-fGA is compared with that of the original p-fGA empirically.

5.1 Performance of the p-fGA and the mp-fGA

The performance of the p-fGA and the mp-fGA were tested on the following two test functions.

(i) *FMS (Frequency Modulation Sounds) parameter identification problem:* f_{fms} . This problem was used to evaluate the p-fGA in [9] and the objective is to determine the 6 parameters (a_1, w_1, a_2, w_2, a_3, w_3) of the FM sound model represented by

$$y(t) = a_1 \sin(w_1 t\theta + a_2 \sin(w_2 t\theta + a_3 \sin(w_3 t\theta))), \tag{1}$$

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:

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

where model data are given by the following equation:

$$y_0(t) = 1.0 \times \sin(5.0 t\theta - 1.5 \times \sin(4.8 t\theta + 2.0 \times \sin(4.9 t\theta))). \tag{3}$$

Each parameter is represented by an 8- bit Gray code in the range - 6.4 to 6.35 and a resolution of 0.05 is used. The total length of a string is 8×6 = 48 bit.

(ii) *Modified Griewank Function:* f_{Giewank} [7]. The function is defined as follows:

$$f_{\text{Griewank}} = \sum_{i=1}^{5} x_i^2 / 4000 - \prod_{i=1}^{5} \cos(x^i / \sqrt{i}) + 1 . \tag{4}$$

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

Maximum number of trials were set to 200,000 for both of the f_{fms} & f_{Griewank} , 30 simulations were done 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 α = 0.05, normal mutation rate P_{nm} = 0.02, high mutation rate P_{hm} = 0.2, K_H =60 and BS_{ratio} =1:1 were commonly used for these experiments. The parameters K_R = 0.8 & 0.3 were used for the functions f_{fms} & $f_{Griewank}$, respectively. Two point crossover operator was applied.

We evaluated these 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). Fig. 5 shows #OPT for restricted number of trials. Table 1 summarizes the results after 200,000 trials. The mp-fGA performed better than the p-fGA for both of the functions f_{fms} & $f_{Griewank}$. For the function f_{fms}, the mp-fGA found global optimal 30 times and the p-fGA found 28 times. Further, MNT of the mp-fGA (40,559.3) is smaller than that of the p-fGA (49,917.1). Similar trend was also seen for the function $f_{Griewank}$(#OPT=28, MNT = 66,099.7 for the p-fGA; #OPT=30, MNT=46,428.6 for the mp-fGA). Thus, the mp-fGA enhances the performance by reducing the MNT (by approximately 20-30 %).

The performance improvement of the mp-fGA can be explained from the #OPT/C (number of runs which found the global optimum in the child populations). For the function f_{fms}, #OPT/C of the p-fGA = 12, #OPT/C of the mp-fGA = 16. Similar results were also found for the function $f_{Griewank}$ (#OPT/C=21 for the p-fGA; #OPT/C = 23 for the mp-fGA). Thus, the mp-fGA found the global optimum more number of times in the child populations than that of the p-fGA; and thus required less number of trials for detecting the global optimum. Results with other hypercube sizes also corroborated the earlier finding.

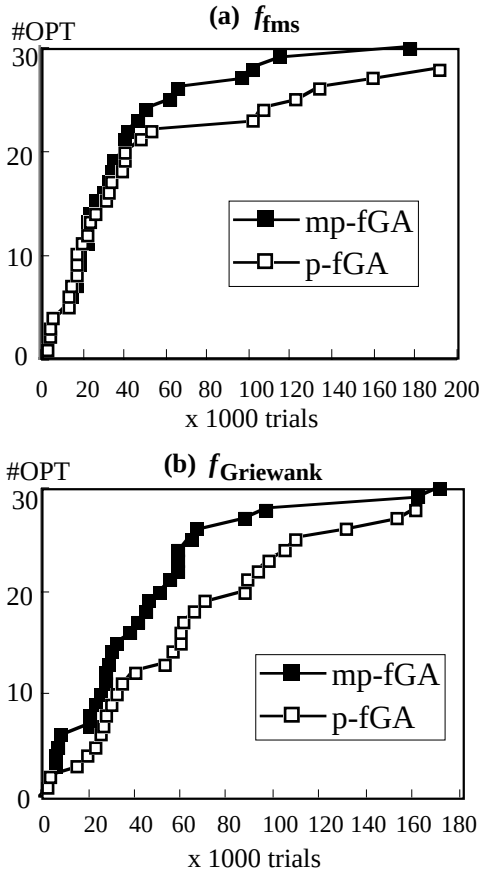


Fig. 5 #OPT for restricted number of trials

Table 1 The mp-fGA vs. the p-fGA

GA \ Function		f_{fms}	$f_{Griewank}$
mp-fGA	**#OPT**	**30**	**30**
	MNT	**40,559.3**	**46,428.6**
	#OPT/C	**16**	**23**
p-fGA	**#OPT**	**28**	**28**
	MNT	**49,917.1**	**66,099.7**
	#OPT/C	**12**	**21**

5.2 Result for the Variable Resolution Method

In this experiment, we tested the vp-fGA with moving windows (mvp-fGA). As reported earlier [9] the following ripple type function is suitable for testing the variable resolution model. Thus, we took

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

where, each x_i is in the range of $0.0 \le x_i \le 100.0$, i = 1, 2, .., 5. The function f_{ripple} has many main peaks of different sizes surrounded by high frequency of small peaks and have the maximum value at $x_1 = x_2 =$, .., $x_5 = 0.1$ with functional value 5.5. 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 parameter K_R = 0.5 and BS_{ratio} = 2:1 were used. Each run continued until the global optimum was found or a maximum of 100,000 trials was reached. Other parameters were the same as in Section 5.1. Coding conditions are as follows: the neighborhood hypercube size s_i = 1.5 is used for each i; to represent each parameter x_i , 12 bits and 11 bits were used in the parent and the child populations, respectively. Thus, the resolution Δx_i of the parent and the

child populations were 0.02442 (= 100.0/(2^{12}-1)) and 0.0000723(= 0.15/(2^{11}-1)), respectively.

#OPTs of the mvp-fGA and the vp-fGA were both 30 (100%). With respect to MNT, the mvp-fGA showed higher performance (17,743.4) than that of the vp-fGA (25,564.6). Fig. 6 shows #OPT for restricted number of trials for both the mvp-fGA and the vp-fGA. Thus, it is evident that the concept of moving window is also effective for the vp-fGA.

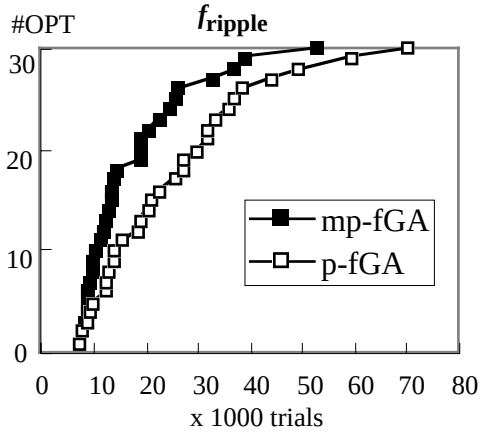


Fig. 6 #OPT for restricted number of trials

6 Conclusions

The concept of phenotypic forking GA (p-fGA) is extended in this article. In the original p-fGA, a neighborhood hypercube or a search sub-space was defined around the best individual at the time of forking with the then best solution as the center of that hypercube. In the present work, we extend this concept by moving the center of the neighborhood hypercubes with generations. Here, the position of the center of the hypercube is updated every generation such that it becomes the current best solution; thereby dynamically modifying the sub-spaces for exploration and exploitation. This enhances the scope of tracing the optimum solution and gives more flexibility for choosing the size of the hypercube. Empirical results on complex function optimization problems show that the new method finds the global optimum in less (by approximately 20-30%) number of trials than the original p-fGA.

Acknowledgments

This research is partially supported by the Ministry of Education, Science, Sports and Culture under Grant-in-Aid for Scientific Research on Priority Areas number 264-08233105.

References

[1] K. Deb and D. E. Goldberg "An investigation of niche and species formation in genetic function optimization," *Proc. Third Inter. Conf. on Genetic Algorithms*,1989, pp. 42-50.

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

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

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

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

[6] K. Mathias and D. Whitley "Changing representations during search: a comparative study of delta coding," *Evolutionary Computation*, vol. 2, no. 3, pp. 249-278, 1994.

[7] A. Torn and A. Zilmskas, *Global optimization* (p. 186). Berlin: Springer-Verlag (Lecture Notes in Computer Science), 1989.

[8] S. Tsutsui and Y. Fujimoto, "Forking genetic algorithm with blocking and shrinking modes," *Proc. Fifth Inter. Conf. on Genetic Algorithm*s, 1993, pp. 206-213.

[9] S. Tsutsui and Y. Fujimoto "Phenotypic forking genetic algorithms," *Proc. IEEE Inter. Conf. on Evolutionary Computation*, 1995, pp. 566-572.

[10] D. Whitley, "Fundamental principles of deception in genetic search," *Foundations of Genetic Algorithms*, pp. 221-241, San Mateo, CA: Morgan Kaufmann, 1991.