Research Article

A Fast and Efficient Genetic Evolution Algorithm

[1]Yu-Cheng Liu and [2]Yu-Bin Liu
[1]School of Electrical and Information Engineering, Chongqing University of Science and Technology, Chongqing, China
[2]School of Continuing Education, Panzhihua University, Panzhihua, China

Abstract: This study presents an improved genetic algorithm. The algorithm introduced acceleration operator in the traditional genetic algorithm, effectively reducing the computational complexity. The search speed of the algorithm has been greatly improved, so that it can quickly find the global optimal solution. The accelerating collaborative operator lessons from the thoughts of binary search algorithm combining with the variable step length strategy. The accelerating operator has strong local search ability and crossover and mutation operators have strong global search ability, then combining these operators generates a new Genetic algorithm. The tests on the different functions show that the improved algorithm has the advantages of faster convergence and higher stability in the case of a small population than traditional genetic algorithm and can effectively avoid the premature phenomenon.

Keywords: Accelerating operator, improved genetic algorithm, search speed, traditional genetic algorithm

INTRODUCTION

As a bionic algorithm in the macro sense. Genetic Algorithm (GA) inspires a good structure by simulating the Darwinian "survival of the fittest, survival of the fittest" principle. It maintains the existing structure and looks for a better structure by simulating Mendelian theory of the genetic variation in the iterative process. The classical genetic algorithm solving steps are initialize population, choice, cross, variation and the judge of the termination condition. The genetic algorithm provides a common framework of solving traditional optimization problems. Because it does not depend on the specific problem areas and has good robustness, it is widely used in many disciplines. With further research, the genetic algorithm also showed many deficiencies, such as premature convergence, easy to fall into local optimum, the slow search speed and strong dependence on the initial population. Because of it's insufficient and inspire of the imitation of human intelligence (HSIC), People have been proposed many improvements algorithm and the new intelligent algorithms, such as the parallel genetic algorithm based on fixed-point theory (Chen *et al*., 2010), adaptive genetic algorithm, super-selection strategy genetic algorithm, chaos genetic algorithm, ant colony algorithm, PSO algorithm, simulated annealing algorithm, immune algorithm (Gong *et al*., 2008) co-

evolutionary algorithm and so on. This study introduced the acceleration operator in the traditional genetic algorithm to enable search efficiency to be greatly improved and to enable the global optimum to be quickly found.

GENETIC ALGORITHM WITH ACCELERATE OPERATOR

This algorithm uses a binary encoding, random initialization population and the elite retention policy. The acceleration operator is applied to each individual. All genes of all individuals in the population are executed crossover and mutation. When the end conditions are satisfied, genetic manipulation ends, or go into the next generation.

Population initialization:
Binary encoding and decoding: To transformer the variable value into the corresponding binary string.

Set $x = [x_1, x_2, \ldots, x_n]^T \varepsilon R^n$ to be decision-making variable, set $u = [u_1, u_2, \ldots, u_n]^T$ to be the upper bound of the decision-making variable, set $d = [d_1, d_2, \ldots d_n]^T$ to be the lower bound of the decision-making variable, set $p = [p_1, p_2, \ldots, p_n]^T$ to be the accuracy of the decision-making variable, set $l = [l_1, l_2, \ldots, l_n]^T$ to be the length of the binary string with corresponding to decision-making variable. The formula is as follows:

Corresponding Author: Yu-Cheng Liu, School of Electrical and Information Engineering, Chongqing University of Science and Technology, University Town, Shapingba, Chongging 401331, P.R. China, Tel.: 13436087448

$$
\begin{aligned}
&t = u_i - d_i; \\
&l_i = 0; \\
&while(t >= p_i) \\
&\{ \\
&\qquad l_i + +; \\
&\qquad t / = 2; \\
&\}
\end{aligned}
\tag{1}
$$

Here, $i \in \{1, 2,.., n\}$. Use $o = [o_1, o_2,\ldots, o_n]^T$ as the lower bound of the binary string with corresponding to decision-making variable, $o_i = 0, i \in \{1,2,\cdots,n\}$. Use u^b $[u^b_1, u^b_2,\ldots, u^b_n]^T$ as the upper bound of the binary string with corresponding to decision-making variable, $u^b_i = 2^{l_i}, i \in \{1,2,\cdots,n\}$. Use $c = [c_1, c_2,.., c_n]^T$ as the gene with corresponding to decision-making variable. The binary string in the machine is expressed as unsigned integer number that itself is a binary string. The decoding formula is as follows:

$$
x_i = d_i + c_i * (u_i - d_i) / u^b_i, i \in \{1,2,\cdots,n\}
\tag{2}
$$

The initialization of the population has a variety of methods, such as random initialization, uniform initialization and orthogonal initialization. In order to show the superiority of the introduction of the acceleration operator, the study selected randomly initialized population. Another advantage of using random initialization is that the number of variables can be arbitrarily changed, which brings the convenience in the program tests. This population size is set to 50.

Acceleration operator:
Seek to the initial exploration step length: Use vector $s^b[s^b_1, s^b_2,.., s^b_n]^T$ to express the exploration step length of each dimension binary string. Assume s^b_i is as follows:

$$
s^b_i = U_i / c, \qquad i \in \{1,2,\cdots,n\}
\tag{3}
$$

Here, c is an appropriate constant. In this study, c = 10. Use $s = [s_1, s_2,\ldots, s_n]^T$ to express the step length of the corresponding variable after decoding. The calculate formula is shown as follows:

$$
s_i = (u_i - d_i) / c, \qquad i \in \{1,2,\cdots,n\}
\tag{4}
$$

The description of using variable was carried out as follows. It's just a difference of a decoder compared with the binary description.

Variable step size strategy: An individual in the population is equivalent to a point in the solution space. It can be expressed as the vector $a = [a_1, a_2,\ldots, a_n]^T$. Select the dimension i, $a^i = [a_1, a_2, \cdots, a_{i-1}, a_i + s_i, a_{i+1}, \cdots, a_n]^T$, fix the other dimension. Also set up a counter times to record number of times of that the fitness value of the point a^i is continuously smaller than the fitness value of the point a. Then determine the next step length

according to the current fitness value of a and a^i and times. The formula for calculating step length is shown as follows:

Sub-histogram after FCM. Therefore we define two weights according to the information. One, W_1, is the ratio of the intensity distribution of each cluster and the whole image. The other one, W_2, is the ratio of the pixel number of each cluster and the whole image. And the total weight, Weight, is composed of W_1 and W_2 and the intensity distribution owns more weight than the pixel number. So we can calculate the weights according to the followings:

$$
s_i = \begin{cases}
s_i; & f(a^i) < f(a) \ \text{and} \ times \leq 2 \\
2 * s_i; & f(a^i) < f(a) \ \text{and} \ times > 2 \\
-s_i / 2; & f(a^i) > f(a) \\
s_i / 2; & f(a^i) = f(a)
\end{cases}
\tag{5}
$$

Here, f indicates the fitness function. What is calculated in the study is the minimum. The corresponding changes are necessary when calculating the maximum.

Steps of the acceleration operator: With having the above initial step length and the calculating formula of the variable step length, we first give the search operator in the i-th dimension before specific steps of the acceleration operator are given.

Operator 1:
The first step: to calculate $a^i = [a_1, a_2, \cdots, a_{i-1}, a_i + s_i, a_{i+1}, \cdots, a_n]^T$ according to step $s = [s_1, s_2,\ldots, s_n]^T$, given point $a = [a_1, a_2,\ldots, a_n]^T$ and the selected i-th dimension.

The second step: If $f(a^i) \leq f(a)$, $a = a^i$, then according to (5) determine the step length of the new i-th dimension step length s^i, $a^i = [a_1, a_2, \cdots, a_{i-1}, a_i + s_i, a_{i+1}, \cdots, a_n]^T$.

The third step: If $s^i = 0$ then end, otherwise turn to the second.

With having the operator 1, we can give the concrete steps of the acceleration operator as follow:

The first step: To determine the initial step length $s = [s_1, s_2,\ldots, s_n]^T$ and starting point $a = [a_1, a_2,\ldots, a_n]^T$, take i = 1.

The second step: To apply operator 1 to the i-th dimension.

The third step: If $i > n$ then end, otherwise $i = i+1$, turn to the second step.

Crossover: There are many the binary crossing ways, such as a single-point crossover, multi-point crossover, uniform crossover, multi-point orthogonal crossover, ectopic crossover (Zhong *et al.*, 2003), multi-agent crossover (Pan and Wang, 1999) and so on. The ectopic crossover among them will change the original model
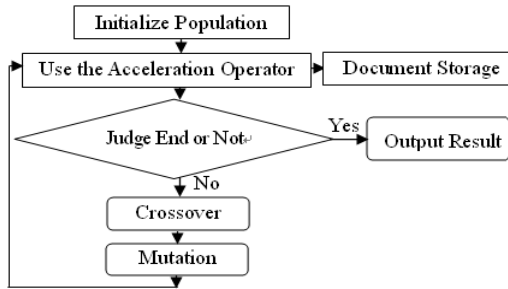
Fig. 1: Algorithm process of genetic algorithm with acceleration operator

space, while others will not change the original model space. The study has taken single-point random crossover manner. The crossover point is randomly generated. Because of the efficiency of the acceleration operator, after the applying the acceleration operator to an individual, one local extremum will be searched. If the acceleration operator is applied to the individual once again, the same local extremum will still be searched. Therefore, this individual does not have much conservation value. Therefore, this study used a simple single-point crossover. All individuals were carried out the crossover. By means of using the offspring individual to replace the parent individual, the global search capability was improved as much as possible.

Here is a concrete example, consider the following 11-bit string length parent individuals.

Parent individual 1: 0 1 1 1 0 0 1 1 0 1 0
Parent individual 2: 1 0 1 0 1 1 0 0 1 0 1

Assuming crossover point position randomly generated to be 5, after the crossover two offspring individuals were generated as follow:

Offspring individual 1: 0 1 1 1 0 0 0 0 1 0 1
Offspring individual 2: 1 0 1 0 1 1 1 1 0 1 0

Mutation: The most basic operation of the binary mutation is to change loci. On this basis, then according to different factors, loci were carried out the mutation so as to have the different mutation algorithms (Liu *et al.*, 2003). The study used the basic mutation. According to a certain mutation rate, mutation position was randomly produced. The bit is inverted. Consider the following 11-bit string length parent individuals:

Parent individual 1: 0 1 1 1 0 0 1 1 0 1 0

Assuming crossover point position randomly generated to be 3, the new offspring individual was generated after the mutation:

Offspring individual 1: 0 1 1 1 0 0 1 1 1 1 0

Genetic algorithm steps with acceleration operator:
The first step: To initialize population.

The second step: To apply the acceleration operator to the individuals in the population. Put the best individual searched into a single document. If the closing conditions is met, the iteration ends, otherwise turn to the third step.

The third step: Crossover.

The fourth step: Mutation.
The algorithm process was shown in Fig. 1.

SIMULATION TEST RESULT ANALYSIS

For the genetic algorithm introduced the acceleration operator, we applied random initialization and taken the population size to be 50, 50 individuals were carried out random single-point crossover and random single-point mutation, the document storages the optimal individual. For the traditional genetic algorithm, we applied random initialization and taken the population size to be 200, 200 individuals were carried out random single-point crossover with having the crossover rate of 25% and random single-point mutation with having the mutation rate of 5%, the championship selection, the document storages the optimal individual. Below are test results of the six categories of the benchmark test function. The number of variables is 3.

The test of the first type test function: The first type test function:

$$f_1(x) = \sum_{i=1}^{3} x_i^2 - 450 \,,\, x_i \in [-100,100]$$

the optimal point $x = (0,0,0)^T$, the optimal value is -450. The new algorithm run 10 generations with taking 92 ms. While the traditional algorithm run 100 generations with taking 2498 ms. Set n to be the number of generations. Set OVS to be optimal value searched. The results of two algorithms were shown in Table 1. The test results showed that the new algorithm found the global optimal value in the third generation, while traditional algorithm found the global optimal value in the forty generation.

The test of the second type test function: The second type test function:

$$f_2(x) = \max_i\{| x_i |,1 \le i \le 3\} - 450 \,,\, x_i \in [-100,100]$$

the optimal point $x = (0,0,0)^T$, the optimal value is -450. The new algorithm run 20 generations with taking 156 ms. While the traditional algorithm run 100 generations with taking 2403 ms. The results of two algorithms were shown in Table 2.
The test results showed that the new algorithm found the global optimal value in the eighteenth generation, while traditional algorithm found the global optimal value in the ninety generation.

Table 1: Test results of the first type test function

	New algorithm					Traditional algorithm			
n	x_1	x_2	x_3	OVS	n	x_1	x_2	x_3	OVS
1	0	1.7578	0.00019	-446.9099	10	3.10379	14.7971	0.06679	-221.413697
2	01953	0.0488	0	-446.9601	20	0	1.95309	0	-446.18538
3	0	0	0	-450.0000	30	0	1.75779	0.00019	-449.08681
4	0	0	0	-450.0000	40	0	0.19531	0	-449.96189
5	0	0	0	-450.0000	50	0	0.04859	0	-449.99761
6	0	0	0	-450.0000	60	0	0.03659	0.00320	-449.99869
7	0	0	0	-450.0000	70	0	0.03659	0.00320	-449.99869
8	0	0	0	-450.0000	80	0	0.03659	0.00320	-449.99869
9	0	0	0	-450.0000	90	0	0.03659	0.00320	-449.99869
10	0	0	0	-450.0000	100	0	0.03359	0	-449.99891

Table 2: Test results of the second type test function

	New algorithm					Traditional algorithm			
n	x_1	x_2	x_3	OVS	n	x_1	x_2	x_3	OVS
2	-1.80029	1.60391	-0.1995	-448.1998	10	0	2.7339	0	-447.26559
4	-1.80029	1.60391	-0.1995	-448.1998	20	0	2.34371	0	-447. 65628
6	-1.80029	1.60391	-0.1995	-448.1998	30	0	0.78128	0.00609	-449.21869
8	-1.80029	1.60391	-0.1995	-448.1998	40	0.08109	0.19529	0.00609	-449.80471
10	-0.92918	-0.7129	-0.5693	-449.0709	50	0.08109	0.19529	0.00609	-449.80471
12	-0.92918	-0.7129	-0.5693	-449.0709	60	0.02457	0.19529	0.00609	-449.80471
14	-0.92918	-0.7129	-0.5693	-449.0709	70	0.02521	0.15682	0.01831	-449.84319
16	-0.92918	-0.7129	-0.5693	-449.0709	80	0.02439	0.12208	0.01831	-449.87789
18	0.0002	0	0^{+3}	-449.9999	90	0.00189	0.02248	0.00609	-449.97751
20	0.0002	0	0^{+3}	-449.9999	100	0.00081	0.01029	0	-449.98981

Table 3: Test results of the third type test function

	New algorithm					Traditional algorithm			
n	x_1	x_2	x_3	OVS	n	x_1	x_2	x_3	OVS
3	-0.999	-1.000	-1.000	391.9985	10	1.56171	7.17769	65.33199	734.36851
6	-1.000	-0.999	-1.000	391.9983	20	2.01719	7.19999	65.34331	607.94668
9	-0.994	-1.000	-1.000	391.9883	30	2.01719	7.19999	65.34331	607.94668
12	-0.994	-1.000	-1.000	391.9883	40	2.01719	7.19999	65.34331	607.94668
15	-0.994	-1.000	-1.000	391.9883	50	2.01719	7.19598	65.34309	597.31498
18	-0.994	-1.000	-1.000	391.9883	60	2.01951	7.06811	64.16021	554.57039
21	-0.994	-1.000	-1.000	391.9883	70	2.01951	7.15339	65.33199	540.30489
24	0.003	0.0024	0.0048	390.0012	80	1.97068	7.12892	65.33199	499.56871
27	0.003	0.0024	0.0048	390.0012	90	1.97068	7.13808	65.33199	493.09489
30	0.003	0.0024	0.0048	390.0012	100	1.95849	7.15329	65.33199	483.01942

Table 4: Test results of the fourth type test function

	New algorithm					Traditional algorithm			
c	x_1	x_2	x_3	OVS	n	x_1	x_2	x_3	OVS
1	0.99	0.9949	1.0322	-326.7360	1	0.04999	1.0099	0.16011	-323.79561
2	0	0.0024	1.0054	-328.9823	5	0.00951	1.05219	0.97499	-327.26818
3	0	0	0	-330	10	0.02081	1.00591	0.97661	-327.83389
4	0	0	0	-330	20	0	0	0.97661	-328.93838
5	0	0	0	-330	30	0	0	0.97661	-328.93838
6	0	0	0	-330	45	0	0	0.97661	-328.93838
7	0	0	0	-330	55	0	0	0.97661	-328.93838
8	0	0	0	-330	70	0.00021	0	1.00679	-328.97721
9	0	0	0	-330	85	0	0	0.99760	-329.00369
10	0	0	0	-330	100	0.00059	0.00151	0.99619	-329.00418

The test of the third type test function: The third type test function:

$$f_3(x) = \sum_{i=1}^{3} (100((x_i+1)^2 - (x_{i+1}+1))^2 + x_i^2) + 390, x_i \in [-100,100]$$

the optimal point $x = (0,0,0)^T$, the optimal value is 390.

The new algorithm run 30 generations with taking 246 ms. While the traditional algorithm run100 generations with taking 2545 ms. The results of two algorithms were shown in Table 3. The test results showed that the new algorithm found the global optimal value in the twenty-fourth generation, while the search results of the traditional algorithm were still quite far away from the global optimal solution and had fallen into a local optimum.

Table 5: Test results of the fifth type test function

New algorithm					Traditional algorithm				
n	x_1	x_2	x_3	OVS	n	x_1	x_2	x_3	OVS
1	0.002	0.0001	10.8665	-179.970	1	310.05	315.6	304.7999	-107.40389
2	-6.270	4.4272	-5.4332	-179.978	10	134.84	187.5	403.2351	-124.27819
3	-6.270	4.4272	-5.4332	-179.978	25	12.887	0.149	304.1439	-156.72501
4	-6.270	4.4272	-5.4332	-179.978	40	14.063	0	4.394498	-178.88438
5	-6.270	4.4272	-5.4332	-179.978	50	12.360	4.688	4.504401	-178.98741
6	-6.270	4.4272	-5.4332	-179.978	60	18.749	0	11.42558	-179.82608
7	-6.270	4.4272	-5.4332	-179.978	70	12.451	0	10.83981	-179.92488
8	-3.135	0	-5.4332	-179.990	80	12.579	0.042	10.83958	-179.93021
9	0	-0.0001	-5.0002	-180	90	12.579	0	10.83979	-179.93068
10	0	-0.0001	-5.0002	-180	100	12.561	0.0023	10.83971	-179.93089

Table 6: Test results of the sixth type test function

New algorithm					Traditional algorithm				
n	x_1	x_2	x_3	OVS	n	x_1	x_2	x_3	OVS
1	-0.9905	0.0001	0.0001	-137.83689	10	0.0627	0.8750	2.0780	-133.64539
2	-0.0002	0.0001	0	-139.99951	20	0.0469	0.1875	0.1875	-138.44489
3	-0.0002	0.0001	0	-139.99951	30	0.0469	0.1875	0.1875	-138.44489
4	-0.0002	0.0001	0	-139.99951	40	0	0.1836	0.0938	-138.91868
5	-0.0002	0.0001	0	-139.99951	50	0	0.1875	0.0625	-138.98771
6	-0.0002	0.0001	0	-139.99951	60	0	0.1757	0.0312	-139.12238
7	-0.0002	0.0001	0	-139.99951	70	0	0.0586	0.0312	-139.77029
8	-0.0002	0.0001	0	-139.99951	80	0.0078	0.0428	0.031	-139.82671
9	-0.0002	0.0001	0	-139.99951	90	0	0.0428	0.0154	-139.85868
10	-0.0002	0.0001	0	-139.99951	100	0	0.0352	0.0073	-139.89422

The test of the fourth type test function: The fourth type test function:

$$f_4(x) = \sum_{i=1}^{3}(x_i^2 - 10\cos(2\pi x_i) + 10) - 330 \ , x_i \in [-5,5]$$

the optimal point $x = (0,0,0)^T$, the optimal value is -330.

The new algorithm run 10 generations with taking 118 ms. While the traditional algorithm run100 generations with taking 2498ms. The results of two algorithms were shown in Table 4.

The test results showed that the new algorithm found the global optimal value in the third generation, while the traditional algorithm did not still find the global optimal value in the hundredth generation.

The test of the fifth type test function: The fifth type test function:

$$f_5(x) = \sum_{i=1}^{3}\frac{x_i^2}{4000} - \prod_{i=1}^{3}\cos(\frac{x_i}{\sqrt{i}}) - 179 \ , \ x_i \in [-600,600]$$

the optimal point $x = (0,0,0)^T$, the optimal value is -180. The new algorithm run 10 generations with taking 102 ms. While the traditional algorithm run100 generations with taking 3025 ms. The results of two algorithms were shown in Table 5. The test results showed that the new algorithm found the global optimal value in the ninth generation, while the traditional algorithm felled into a local optimum and could not come out.

The test of the fourth type test function: The sixth type test function:

$$f_6(x) = -20\exp(-0.2\sqrt{\frac{1}{3}\sum_{i=1}^{3}x_i^2}) - \exp(\frac{1}{3}\sum_{i=1}^{3}\cos(2\pi x_i)) + e - 120$$

$$x_i \in [-32,32]$$

the optimal point $x = (0,0,0)^T$, the optimal value is -140. The new algorithm run 10 generations with taking 100 ms. While the traditional algorithm run100 generations with taking 1384 ms. The results of two algorithms were shown in Table 6.

The test results showed that the new algorithm found the global optimal value in the second generation, while the traditional algorithm found the global optimal value in the seventieth generation.

CONCLUSION

The test results showed that the new algorithm had successfully passed the test of six types of the test functions. From the running time and the running results, the new algorithm is superior to the traditional algorithm. The main characteristics of the genetic algorithm introduced the acceleration operator are as follows:

- The small population
- The global search is separated from the local search, the crossover and mutation only need to search the local area of containing global optimal solution. The optimal solution in the local area is completed by the acceleration operator

- The search speed is fast, the search efficiency is high

REFERENCES

Chen, X.S., D. Liang and H.Y. Wang, 2010. Artificial fish swarm algorithm with the integration of genetic algorithm for solving the problem of clustering. Anhui Agric. Sci., 38(36): 21068-21071.

Gong, M.G., L.C. Jiao and H.F. Du, 2008. Multiobjective immune algorithm with nondominated neighbor-based selection. Evol. Comput., 16(2): 225-255.

Liu, Z.M., J.L. Zhou and L. Chen, 2003. The research on the mutation operator of genetic algorithm for maintaining the diversity of the population. Small-Scale Micro-Comput. Syst., 24(5): 902-904.

Pan, D. and A.L. Wang, 1999. The genetic algorithm for many individuals to participate the crossover. J. Shanghai Jiaotong Univ., 33(11): 1453-1457.

Zhong, G.K., B. Zhen and Y.Q. Yu, 2003. Genetic algorithm based on ectopic crossover. Control Decision, 18(3): 361-363.