A Novel and Accelerated Genetic Algorithm

HUANG BAO-JUAN, ZHUANG JIAN, YU DE-HONG
School of Mechanical Engineering
Xi’an Jiaotong University
710049, Xi’an
CHINA
bj_huang8@163.com

Abstract: - Genetic algorithm (GA) is very helpful when the developer does not have precise domain expertise, because GA possesses the ability to explore and learn from their domain. At present, the research of GA mainly focuses on the three operators and devotes to improve the algorithm efficiency and avoid premature convergence. This paper presents a cycle mutation operator and a novel selection operator; accordingly, an improved cycle mutation genetic algorithm (ICMGA) is schemed, The experimental results compared with other genetic algorithms validate the performance of this algorithm, such as the exploration ability in search space, the stabilization and calculation speed, are all superior to other algorithms, and ICMGA is not sensitive to the initial population distribution.

Keywords: - Genetic algorithm, Evolution algorithm, mutation operator, selection operator, and cycle mutation operator

1 Introduction

Genetic algorithm is a branch of the evolutionary algorithms which has been established based upon the “survival of the best” and “the proliferation of the superior species” as inspired by Darwin evolutionary hypothesis [1]; it involves three types of operators: selection, crossover, and mutation. By some rules, the selection operator chooses those individuals in the population that will transmit their genes to next generation. The crossover exchanges partial genes of two chosen individuals to create the new offspring that inherit some characters of their parents. A crossover operator manipulates a pair of individuals (called parents) to produce two new individuals (called offspring) by exchanging segments from the parents’ coding. By exchanging information between two parents, the crossover operator provides a powerful exploration capability. A commonly used method for crossover is called one-point. Mutation randomly flips one or more bits in an individual and mutates the bits to create new offspring. The offspring created by mutation may have some new characters or traits that not found in the parent.

Genetic Algorithms can be described as:

1) Choose initial population
2) Evaluate the fitness of each individual in the population
3) Repeat:
   a) Select best-ranking individuals to reproduce
   b) Breed new generation through crossover and mutation (genetic operations) and give birth to offspring
   c) Evaluate the individual fitness of the offspring
   d) Replace worst ranked part of population with offspring
4) Until termination

GA can process complex questions and finds the optimal or near optimal solutions [1] [2] [3] [4], Genetic Algorithms has been widely studied, experimented and applied in many fields in engineering worlds. Not only does GA provide an alternative method to solving problem, it consistently outperforms other traditional methods in most of the problems link. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GA. Its usefulness and gracefulness of solving problems has made it the more favorite choice among the traditional methods, namely gradient search, random search and others. GA is very helpful when the developer does not have precise domain expertise, because GA possesses the ability to explore and learn from their domain. At present, the research of GA mainly focuses on the three operators and devotes to improve the algorithm efficiency and avoid premature convergence.

2 Cycle mutation operator

Mutation can escape the local search and increase the probability of finding global optimal solutions. Nevertheless, the overlarge mutation probability may lead GA to random search, thereby reducing the searching efficiency of algorithm. The existing mutation operators can be approximately divided into two species: fixed probability and variable probability. Through results of previous research it can be concluded that the latter excels the former in performance.

The mutations with dynamic probability are classified into three types in [5]: dynamic parameter control, adaptive parameter control and self-adaptive parameter control. In dynamic parameter control, the relationship between mutation probability and generation number usually accords with deterministic decreasing function. Fogarty [6] does a lot of researches and experiments, and finally gives the empirical exponential functional relationship:

$$p_m(t) = \frac{1}{240} + \frac{0.11375}{2^t} \tag{1}$$

Reference [7] gives the more universal functional relationship:

$$p_m(t) = \sqrt{\frac{c_1}{c_2}} \frac{\exp(-c_3 t/2)}{n\sqrt{l}} \tag{2}$$

Here l is the length of individual gene, n is the population size.

Unfortunately, the value of coefficient C_i in equation (2) should be estimated according as the practical problem. Reference [8] gives the much better empirical formula. And its performance is indicated by the test results of a great deal of difficult combinatorial optimization problems.

$$p_m(t) = (2 + \frac{l-2}{T-1}t)^{-1} \tag{3}$$

Here T is the generation number.

The adaptive parameter control adjusts the size of mutation probability according to the information of searching results returned from optimization procedure. An early example of this method is Rechenberg's 1/5 success rule [9]. It requests that at least 20% of the generations are successful after mutations or else increase the mutation probability.

The self-adaptive parameter control has no feedback mechanism that can direct control the value of mutation parameter. But it accords the rule that individual whose mutation parameter values taking on higher performance has evolution dominance and will be allowed to proliferate in population. There are many successful applications of self-adaptive parameter control in continuous optimization of evolution strategy and evolution programming [10] [11].

In practice, it has been found that premature phenomena appear when the population size is small. To solve this question, literature [12] introduces mutation probability adjusting scheme. But population affinity needs to be calculated in that method, which will increase the computational cost. Therefore, we expect that acquired mutation operator possesses following performance:

a) It can maintain lower mutation probability in a relatively long period. That means the crossover operator plays a leading role in GA. Consequently, the algorithm almost fully searches the space domain which population covered;

b) It can output high mutation probability in given time periods, which enable GA to jump from local search.

c) It can achieve the adjusting of mutation parameter with low computation cost.

To acquire this mutation operator, we centralize our insight on the most successful evolution example —human evolution and analyze the rules that followed by biological evolution. Paleontologist demonstrates through fossil that there appears 5 times biology extinction in the organism's development history, and the cycle of extinction appeared is basically between 62 million years and 65 million years [13]. In each cycle, there exist excellent populations in evolution race, such as the Jurassic dinosaurs. Similarly, there is no extinction in the organism's evolutionary process; there is no appearance of human being. Based on this clue, a mutation operator with cycle probabilities that borrows the evolution experience of earth biology is introduced; the operator is designed as follows:

$$p_m(t) = \frac{\alpha\left[t-(k+0.5)T_c\right]^2}{T_c^2} \tag{4}$$

Here t is generation number, T_c is variety cycle, α is adjustment coefficient of probability, and k is cycle number.

In equation (4) the parameter need to be pre-defined only T_c and α. Generally, the value of T_c is the least expected number of evolution generation, but the value should not be too small. The value of α is the extremum of mutation probability P_m. Reference [14] proves through Markov chain that the GA with elitists reserved is convergent, and literature [15] points out that if GA is ergodic and the operation of elitists reserved can not alter or affect its ergodic property, then the algorithm is global convergent. So that the operation of elitists reserved is added to our algorithm in order to ensure its convergence property.

To illustrate the performance of this mutation, 22 functions are tested and the results are compared between 4 kinds of GA. Each function is tested 1000 times and in each time the number of iteration is most 1000. Furthermore, the parameters of 4 GA are binary encoded and the code length is 20.

Algorithm 1: Classical genetic algorithm (SGA), population size is 50, P_m=0.01 and P_s=0.7;

Algorithm2: GA introduced by Reference [8] (BSGA), in which the probability of mutation can adjust dynamically. Population size is 50, P_m=0.01 and P_s=0.7;

Algorithm3: GA introduced by Dirk Thierens [5] (DMGA), in which the probability of mutation can adjust adaptively. Population size is 50, P_m=0.5, Ps=0.7, λ=1.1, ν=1.0 and ω=1.5;

Algorithm4: Cycle mutation GA of this paper (CMGA), population size is 50, T_c=50 and α=1.0.

Table 1 presented the experimental results. It is obvious that the performance of BSGA, DMGA and CMGA is almost similar to each other, stability and calculating speed of BSGA and CMGA is superior to that of DMGA; but all of them are excels SGA in performance. That shows that the cycle mutation, which simulates the rules of biologic evolution, can be applied to GA and its performance is not inferior to that of other elaborately mutations. But it still not reaches our expectation. What is the key to solve this problem?

Table 1 Performance testing results
(M1: SGA, M2: BSGA, M3: DMGA, M4: CMGA)

	Times of object found				**Mean square deviation**				**Average time cost (ms)**			
	M1	**M2**	**M3**	**M4**	**M1**	**M2**	**M3**	**M4**	**M1**	**M2**	**M3**	**M4**
F1	53	1000	997	1000	14.05	1.36	1.62	0.97	0.51	0.48	1.14	0.46
F2	0	1000	985	1000	/	2.39	3.60	2.27	/	0.48	1.13	0.48
F3	38	1000	969	1000	2.13	1.03	1.35	1.54	0.88	0.43	1.17	0.44
F4	6	1000	934	1000	0.00	1.84	6.60	2.04	0.00	0.44	1.12	0.45
F5	0	998	861	1000	/	1.08	0.88	2.72	/	0.48	1.28	0.49
F6	29	1000	998	1000	31.50	0.85	0.46	0.50	0.64	0.43	1.12	0.42
F7	0	11	150	14	/	66.11	18.82	81.11	/	0.47	1.21	0.48
F8	0	54	0	73	/	39.79	/	30.92	/	0.46	/	0.44
F9	0	0	120	0	/	/	17.30	/	/	/	1.24	/
F10	21	793	339	806	42.97	7.92	10.21	8.66	0.36	0.52	1.06	0.48
F11	16	1000	666	1000	57.70	2.35	2.51	2.84	0.50	0.53	1.08	0.47
F12	180	1000	978	1000	5.56	0.92	1.16	0.89	0.20	0.46	1.02	0.43
F13	38	922	471	994	0.00	4.64	5.35	5.67	0.00	0.46	1.04	0.45
F14	3	1000	976	1000	0.00	1.06	1.59	2.13	0.00	0.52	1.21	0.46
F15	102	1000	995	1000	7.70	1.29	1.80	1.10	0.56	0.52	1.09	0.45
F16	0	0	0	0	/	/	/	/	/	/	/	/
F17	0	845	191	992	/	5.49	4.52	4.98	/	0.48	1.02	0.43
F18	0	5	23	0	/	73.43	56.96	/	/	0.54	1.12	/
F19	23	380	49	602	18.68	6.63	16.06	11.35	0.37	1.00	1.63	1.01
F20	0	0	0	0	/	/	/	/	/	/	/	/
F21	0	466	8	654	/	8.41	3.39	8.83	/	0.48	1.16	0.48
F22	84	1000	998	1000	12.45	0.95	0.34	0.73	0.47	0.50	1.19	0.48

3 Selection operator

Mutation is the motivity for gaining new schemas in GA, but the searching efficiency in local space need to be achieved through selection and crossover operators. Especially the selection, which directly reflects “the Survival of the Fittest” theory of biological evolution, determined the evolutionary direction of GA. Selection are usually classified as three types [16], namely stability selection, direction selection and rupture selection. Stability selection is also called as generalization selection for its trend to remove individuals with extreme fitness, direction selection can increase or decrease the average fitness of population, and rupture selection has the ability of eliminating individuals with middle fitness. No matter what kinds of operator it is, the individuals under operation with higher fitness always have a relatively high probability in transmitting their own genes to next generation. The existing selection operators in GA are mainly composed of proportion selection [17] [18] [19], ranking selection [20] [21], etc.; all of them can be regarded as the evolution of direction selection. Under the restriction of GA population size, these selection operators induce the high fitness individuals producing more offspring in next generation; although that is the basic mechanism of GA to find optimal solutions, it can lead to premature convergence. Therefore, when designing the selection, the best of all attention issue is how to avoid prematurity. In GA, the most frequently used selection is proportion selection (or be called as roulette wheel selection), it can be described as follows:

$$trs(x_i) = \frac{u(x_i)}{\sum_{j=1}^{n} u(x_j)} \quad (5)$$

The proportion selection has the character that the individuals with higher fitness have the higher probability of transmitting their genes to next generation. But practical experiments indicate that the lower fitness ones can also transmit their genes to next generation though proportion selection, and the transfer probability is rather high.

Table 2 shows the statistical ranking results of the elitists' parents, the algorithm used in this experiment is CMGA, population size is 50, T_c=50,α=1.0; Function 1 in appendix1 is selected as cost function and 10 times are tested in all.

Table 2 the ranking result of the elitist' parents in CMGA

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10
Elitist (times)	46	23	23	10	46	22	10	46	23	10
Not-elitist (times)	38	15	15	6	38	16	6	38	15	6

From table 2 it can conclude that the GA with roulette wheel selection whose elitists hold the leading position in evolution process. But the probability of other non-optimum ones being selected as parent is close to 40%. That makes us have to think the rationality of probability.

In order to make the selection more rational, the probability that individual *x* falling into parent is revised:

$$P_{trs}(x) = \frac{0.7u(x)}{u_{\max}(t)} \quad (6)$$

Here $u_{max}(t)$ is the individual with maximum fitness in *t* generation. $u(x)$ is the fitness of individual *x*, $P_{trs}(x,t)$ is the probability that individual *x* being selected as parent in *t* generation.

4 An improved cycle mutation genetic algorithm

Integrated the improved selection operator and CMGA, a new GA named improved cycle mutation genetic algorithm (ICMGA) is introduced as follows:

1. Set the population size *PopuSize*, the probability of crossover operator P_s, the number of reserved elitists *BackupSize*;
2. Initialize population;
3. Count population, reserve the optimum population;

3.1 First reserve the best elitist in generation, and then reserve the *BackupSize*-1 individuals that are most close to the elitist. The distance calculation formula for parameter is defined as follows:

$$d^i = \sum_{k=0}^{pn} \left(Parameter_k^i - Parameter_k^{\max} \right)^2 \quad (7)$$

4. Set the number of iteration *IterativeTimes*=0;
5. Test the stop criterion, if not satisfy, continue; Else, go to 10;
6. Set the number of generated offspring *i*=0;
7. Test *i*=*PopuSize*, if not, continue; Else, go to 8;

7.1 Select two parent individuals' f_i and m_i by the equation (6);

7.2 Perform crossover operation; generate two offspring individuals' son_i and son_{i+1};

7.3 Calculate the probability of mutation operator by the equation (4);

7.4 For every gene of the two offspring individuals, operate mutation in terms of P_s.

7.5 Calculate the individual fitness value of the two offspring individuals;

7.6 *i*=*i*+2, go back to 7;

8. Count the offspring population; update the set of advantage population;

8.1 If there are elitists which fitness value is higher than those in the set of advantage population, substitute these elitists for those in the set and update the other elitists in the scope of offspring and the set according as the principle of nearest; Else, release those in the set into offspring;

9. *IterativeTimes*=*IterativeTimes*+1, go back to 5;
10. Output result, stop the program.

5 Results and discussion

5.1 Comparative test in algorithm performance

To test the performance of ICMGA, the self-adaptation genetic algorithm (AGA) [12] and simple immune clone algorithms (SIA) [22] are used to compare. The basic settings of experiment are listed: the operation system: Window XP, the programming language: C++, the computer memory: 1GB and the dominant frequency of CPU: 3GHz. 22 functions are tested in all, each function is tested 1000 times and in each time the number of iteration is 1000. The parameters of 3 GA are binary encoded and the code length is 20.

Algorithm 1 (M1): SIA, population size is 50 and clone number is 5;

Algorithm 2 (M2): AGA, population size is 50, the number of reserved optimum population is 10, P_s=0.7;

Algorithm 3 (M3): ICMGA, population size is 50, the number of reserved optimum population is 10, P_s=0.7;

The comparison result is showed in Table 3.

Table 3 Comparison results of three methods (M1: SIA; M2: AGA; M3: ICMGA)

	Times of object found			**Mean-square deviation**			**Average iteration number**			**Average time cost (ms)**		
	M1	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**	**M1**	**M2**	**M3**
F1	1000	1000	1000	0.99	1.78	0.57	31.21	81.19	26.34	2.36	0.50	0.48
F2	59	1000	1000	37.54	0.38	0.17	425.93	44.96	23.04	2.84	0.49	0.49
F3	1000	998	998	2.71	5.74	2.19	69.67	117.43	49.65	2.49	0.51	0.51
F4	704	1000	1000	11.13	0.48	0.52	338.99	51.86	30.43	2.64	0.49	0.47
F5	851	905	997	8.76	2.71	3.61	258.04	176.46	108.43	3.13	0.54	0.63
F6	994	1000	1000	4.79	0.75	0.32	131.98	53.17	22.68	2.52	0.50	0.49
F7	66	10	1000	37.04	52.86	0.70	415.29	728.60	43.19	2.84	0.57	0.50
F8	551	90	992	12.64	28.28	5.01	333.05	457.24	206.02	2.63	0.55	0.50
F9	0	5	1000	/	0	0.47	/	18.00	36.34	/	0.51	0.53
F10	1000	799	998	2.06	8.88	4.62	57.22	327.26	95.56	2.30	0.56	0.48
F11	1000	998	1000	1.42	7.59	2.79	43.89	275.67	79.57	2.35	0.51	0.49
F12	1000	1000	1000	0.07	2.03	0.48	4.46	42.55	14.37	1.19	0.43	0.38
F13	1000	1000	1000	0.34	1.36	0.26	11.68	75.17	14.06	1.83	0.52	0.42
F14	700	942	1000	10.86	2.43	2.38	315.40	87.41	71.44	2.98	0.52	0.53
F15	1000	1000	1000	0.78	2.56	0.61	23.88	80.60	26.11	2.07	0.49	0.46
F16	0	0	974	/	/	5.14	/	/	159.22	/	/	0.53
F17	924	1000	1000	8.25	2.55	2.26	239.92	100.35	61.88	3.22	0.74	0.73
F18	0	3	1000	/	0.00	0.20	/	6.00	29.09	/	0.64	0.69
F19	1000	811	991	2.27	8.67	4.03	54.07	310.77	69.57	6.42	1.02	1.02
F20	8	115	762	115.89	32.46	9.15	324.63	235.26	378.44	2.91	0.37	0.53
F21	3	977	980	165.69	9.45	4.95	626.67	203.32	115.42	3.00	0.51	0.50
F22	1000	1000	1000	2.50	0.79	0.48	70.08	37.94	22.99	2.60	0.51	0.52

It can be seen in the table3 that the performance of ICMGA is most excellence. Its probability of finding optimal solutions is almost above 98% (only function 20 that the probability is 76.2%). And for test function F9, F16, F18, F20 and F21, only ICMGA can find the optimal solutions and the finding times more than 760 in 1000 times test. That means the seeking ability for optimal solutions of ICMGA is stronger than other optimization algorithms. The compare results in indexes, such as mean square deviation, average iteration number and average time cost, also shows that ICMGA is better than other two algorithms at stability and calculation velocity.

5.2 The effect test of initial population distribution

To analyze the effect of initial population distribution on performance of ICMGA, four types of binary coding methods are used to code the individuals in initial population. That is 0,1 random uniform distribution, all 1 distribution, all 0 distribution and 0,1 interval distribution, the test environment and method are all as above, the test results are showed in Table 4.

Table 4 Test results of ICMGA initial population in different distribution
(d1: random uniform distribution; d2: all 1; d3: all 0; d4: 0,1 interval)

	Times of object found				**Mean-square deviation**				**Average time cost**			
	d1	**d2**	**d3**	**d4**	**d1**	**d2**	**d3**	**d4**	**d1**	**d2**	**d3**	**d4**
F1	1000	1000	1000	1000	0.57	0.61	0.67	0.56	0.48	0.49	0.48	0.48
F2	1000	1000	1000	1000	0.17	0.00	0.00	0.11	0.49	0.00	0.18	0.49
F3	998	998	1000	1000	2.19	2.86	2.66	2.62	0.51	0.51	0.51	0.51
F4	1000	1000	1000	1000	0.52	0.58	0.55	0.56	0.47	0.42	0.43	0.48
F5	997	998	1000	1000	3.61	3.16	3.58	2.96	0.63	0.62	0.63	0.63
F6	1000	1000	1000	1000	0.32	0.25	0.43	0.50	0.49	0.49	0.49	0.49
F7	1000	1000	1000	1000	0.70	0.60	0.82	0.85	0.50	0.46	0.46	0.49
F8	992	976	994	986	5.01	6.03	5.54	5.75	0.50	0.50	0.50	0.50
F9	1000	1000	1000	1000	0.47	0.40	0.50	0.50	0.53	0.52	0.52	0.53
F10	998	998	1000	984	4.62	4.23	4.02	5.85	0.48	0.41	0.41	0.50
F11	1000	1000	1000	1000	2.79	2.32	2.61	2.62	0.49	0.45	0.43	0.50
F12	1000	1000	1000	1000	0.48	0.45	0.39	0.54	0.38	0.41	0.40	0.41
F13	1000	1000	1000	1000	0.26	0.46	0.38	0.39	0.42	0.44	0.44	0.47
F14	1000	1000	1000	1000	2.38	2.44	2.38	2.39	0.53	0.53	0.53	0.53
F15	1000	1000	1000	1000	0.61	0.83	1.14	0.70	0.46	0.48	0.48	0.46
F16	974	994	994	986	5.14	3.84	4.34	4.69	0.53	0.50	0.55	0.53
F17	1000	1000	1000	1000	2.26	2.01	2.69	3.12	0.73	0.69	0.69	0.69
F18	1000	1000	1000	1000	0.20	0.29	0.29	0.34	0.69	0.65	0.67	0.70
F19	991	997	997	993	4.03	2.48	3.04	4.46	1.02	1.03	1.03	1.04
F20	762	1000	1000	766	9.15	0.84	0.00	8.52	0.53	0.20	0.00	0.53
F21	980	992	992	994	4.96	4.75	4.87	4.30	0.50	0.50	0.50	0.51
F22	1000	1000	1000	1000	0.48	0.46	0.51	0.39	0.52	0.52	0.52	0.53

From the experiment results in table 4, it can draw a conclusion that the initial distribution of population is almost no effect on the performance of ICAMA.

5.3 The performance analysis test

Compared CMGA in table 1 with ICMGA in table 4, it can be seen that the performance of ICMGA is significant superior to that of CMGA. And the only difference between the two algorithms is selection operator. So we will analyze the reason that the selection of ICMGA improves the algorithm performance. In order to detect the difference in effect of selection operator on population evolution, ICMGA and CMGA are iterated 1000 times for the optimization of test function F1 and the times that each individual is selected (according to ranking order) as parent in resultant population are count respectively. For the convenience of data regression analysis, sets the initial population are all parents and population size is 50. And the regression analysis of testing data, ranking order of individuals and the times being selected as parents are implemented by linear function, exponential function and power function respectively. The three equations of regression analysis are listed as follows:

$$\left.\begin{aligned} y &= ax+b \\ y &= ae^{bx} \\ y &= ax^{b} \end{aligned}\right\} \tag{8}$$

Table 5 lists the coefficients of the above three equations and the value of R^2. From the table we can find that the relationship between individuals' ranking order and times being selected as parent in CMGA is mainly consistent with linear equation (for R^2=0.9892), and that in ICMGA accords with power format (for R^2=0.7969)

Table 5 the coefficients of three equations of regression analysis and the value of R2

	CMGA			ICMGA		
	a	b	R^2	a	b	R^2
Linear fitting	-35. 1020	1837.7000	0.9892	-120.2000	4026.4000	0.1098
Exponential fitting	2644.4000	-0.0469	0.8406	61.0880	-0.1190	0.4228
Power fitting	4969.9000	-0.6114	0.5512	7249.2000	-2.6293	0.7969

The results of statistical regression analysis indicated that the population in evolution process accord with Power Law under the selection operator in equation (6). We consider that is the basic reason for high efficiency of ICMGA.

6 Conclusions

In this paper, a mutation operator with cycle probabilities is designed to solve the contradiction of population diversity and over local search. And combined with this mutation and improved selection, a novel and accelerated genetic algorithm (ICMGA) is introduced. The experimental results compared with other genetic algorithms validate the performance of this algorithm, such as the exploration ability in search space, the stabilization and calculation speed, are all superior to other algorithms. Moreover, the statistical analysis of individuals ranking order and times being selected as parent all indicate that the evolution process of ICMGA accorded with Power Law, that is the reason for high performance.

References:

[1] B.Fahimnia, R.Molaei, M.Ebrahimi. Genetic Algorithm Optimization of Fuel Consumption in Compressor Stations, WSEAS TRANSACTIONS on system and control, issue 1, volume 3,January 2008,pp. 1-10

[2] Supachate Innet, Nawat Nuntasen. University Timetabling Using Evolutionary Computation, WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION, Issue 12, Volume 4, Decmber 2007,pp. 243-250

[3] Chen, R.C., Chen, T.S., Feng, C.C., Lin, C.C. and Lin, K.C., Application of Genetic Algorithm on Production Scheduling of Elastic Knitted Fabrics, *Engineering and Applied Sciences,* vol. 1, no. 2, 2006, pp. 149-153.

[4] Edmary Altamiranda,Rodrigo Calderón,Eliezer Colina Morles. An Evolutionary Algorithm for Linear Systems Identification. Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation, Corfu Island, Greece, February 16-19, 2007,pp.225-229.

[5] Dirk Thierens. Adaptive Mutation rate control schemes in genetic algorithms. Evolutionary Computation, 2002. CEC apos; 02. Proceedings of the 2002 Congress on Volume 1, Issue, 12-17 May 2002,pp. 980-985

[6] Fogaty T. Varying the probability of Mutation in the Genetic Algorithm, *Proc. Of the Third International Conference on Genetic Algorithms,* pp. 104-109, Morgan Kaufmann, 1989.

[7] Hesser J. and Manner R. towards an Optimal Mutation Probability in Genetic Algorithms, *Proc, of 1st Parallel Problem Solving form Nature,* pp. 115-124, Springer, 1991.

[8] Back T. and Schutz M. Intelligent Mutation Rate Control in Canonical Genetic Algorithms, *Proc. Of the International Symposium On Methodologies for Intelligent Systems,* pp.158-167, 1996

[9] Rechenberg I. Evolutions strategie: Optimierung technischer Systeme mach Prinzipin der biologischen Evolution, Frommann, 1973.

[10] Schwefel H. -P. Evolution and Optimum Seeking. Wiley, NY, 1995.

[11] Fogel D. Evolutionary Computation: Toward a New Philosophy of machine Intelligence, IEEE Press, NJ, 1995

[12] Zhuang Jian, Wang Sun-an. Study on Self-Adjusting of Gene Migration Genetic Algorithm, Journal *of Xi'an Jiaotong University,* 2002, 36(11), pp.1170~1172.

[13] Rong Jia-yu, Zhan Ren-bin. Re-evaluation of survivors, Lazarus taxa, and refugia from mass extinction, *EARTH SCIENCE FRONTIERS,* 2006, 13(6), pp.187-198.

[14] Eiben A E etc. Global convergence of genetic algorithm: an infinite markov chain analysis In: Schwefel H P, M anner R Eds. Parallel problem solving from Nature. Heidelberg, Berlin: Springer-verlag, 1991, pp. 4-12.

[15] He Lin,Wang Ke-jun. Elitist preserved genetic alorithm and its convergence analysis, *Control and Decision*,2000,15(1) ,pp.63 - 66.

[16] J. J. Grefenstette and J. E. Baker, “How genetic algorithms work A critical look at implicit parallelism,” *in Proc. Third Int. Conf on Genetic Algorithms. San Mateo CA: Morgan*

Kaufmann. 1989, pp, 20-27.

[17] Ting Kuo and Shu-Yuen Hwang. A Genetic Algorithm with Disruptive Selection,*IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B: CYBERNETICS.* 1996, 26(2), pp.299-307.

[18] M. Srinivas, and L. M. Patnaik. Adaptive Probabilities of Crossover Genetic in Mutation and Algorithms.*IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS*, 1994, 24(4), pp.656-667.

[19] J. E. Baker. Adaptive selection methods for genetic algorithms. *In Proc. First Int. Conf on Genetic Algorithms and Their Applications. Hillsdale, NJ Lawrence Erlbaum*, 1985, pp. 101-111.

[20] Vlasis K. Koumousis and Christos P. Katsaras. A Saw-Tooth Genetic Algorithm Combining the Effects of Variable Population Size and Reinitialization to Enhance Performance,*IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 2006, 10(1),pp.19-28.

[21] Chang-Yong Lee. Entropy-Boltzmann Selection in the Genetic Algorithms,*IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS,* 2003, 33(1),pp.138-142.

[22] Liu Ruo-chen, Du Hai-feng,Jiao Li-chen. Immune Monoclonal strategy based on the Cauthy mutation. *Journal of Xidian University,* 2004 31(4), pp.551-556.

Appendix
Test functions and stop qualification:
Function1 **(F1):**

$$\max\ f(x,y)=1+x\times\sin(4\pi x)-y\times\sin(4\pi y+\pi)+\frac{\sin(6\sqrt{x^2+y^2})}{6\sqrt{x^2+y^2+10^{-15}}},\ x,y\in[-1,1]$$

Stop qualification**:** 2.1180

Function2 **(F2):**

$$\max\ f(x,y)=x^2+y^2,\ x,y\in[0,1]$$

Stop qualification**:** 1.9990

Function3 **(F3):**

$$\max\ f(x,y)=3(1-x)^2\exp[-x^2-(y+1)^2]-10(\frac{x}{5}-x^3-y^5)\exp(-x^2-y^2)$$
$$-\frac{1}{3}\exp[-(x+1)^2-y^2],\ x,y\in[-3,3]$$

Stop qualification**:** 8.1040

Function4 **(F4):**

$$\max\ f(x,y)=[\frac{b}{a+(x^2+y^2)}]^2+(x^2+y^2)^2,\ x,y\in[-5.12,5.12],a=0.05,b=3.0$$

Stop qualification**:** 3590.0

Function5 **(F5):**

$$\max\ f(x,y)=\sin(5.1\pi x+0.5)^{30}\exp[\frac{-4\lg 2(x-0.0667)^2}{0.64}]$$
$$\sin(5.1\pi y+0.5)^{30}\exp[\frac{-4\lg 2(y-0.0667)^2}{0.64}],\ x,y\in[0,1]$$

Stop qualification**:** 0.9960

Function 6**(F6):**

$$\max\ f(x,y)=\frac{-1}{2}(x\sin(\sqrt{|x|})+y\sin(\sqrt{|y|})),\ \ x,y\in[-500,500]$$

Stop qualification**:** 418.80

Function7 **(F7):**

$$\max\ f(x,y)=-[20+x^2-10\cos(2\pi x)+y^2-10\cos(2\pi y)],\ \ x,y\in[-5.0,5.0]$$

Stop qualification**:** -0.0010

Function8 **(F8):**

$$\max\ f(x,y)=-[100*(x^2-y)^2+(1-x)^2],\ \ x\in[-8,8]$$

Stop qualification**:** -0.0010

Function9 **(F9):**

$$\max\ f(x,y)=20\exp[-0.2\sqrt{\frac{1}{2}(x^2+y^2)}\,]+\exp\{\frac{1}{2}[\cos(2\pi x)+\cos(2\pi y)]\}-20-e,\quad -10\le x\le 10$$

Stop qualification**:** -0.0010

Function10 **(F10):**

$$\max\ f(x,y)=-[\frac{1}{4000}(x^2+y^2)-\cos(x)\cos(\frac{y}{\sqrt{2}})+1],\ \ x,y\in[-10,10]$$

Stop qualification**:** -0.0010

Function11 **(F11):**

$$\max\ f(x,y)=-0.5-\frac{\sin^2\sqrt{x^2+y^2}-0.5}{[1+0.001\times(x^2+y^2)^2]^2},\ \ x,y\in[-5,5]$$

Stop qualification**:** 0.9990

Function12 **(F12):**

$$\max\ f(x)=10+\frac{\sin(\frac{1}{x+10^{-10}})}{(x-0.16)^2+0.1},\ \ x\in[-10,10]$$

Stop qualification**:** 19.7810

Function13 **(F13):**

$$\max\ f(x,y)=-[x^2+y^2-0.3\times\cos(3\pi x)+0.3\times\cos(4\pi y)+0.3],\ \ x,y\in[-1,1]$$

Stop qualification**:** 0.2390

Function14 **(F14):**

$$\max\ f(x,y)=-\{\sum_{i=1}^{5}i\cos[(i+1)x+i]\}\times\{\sum_{i=1}^{5}i\cos[(i+1)y+i]\},\ \ x,y\in[-10,10]$$

Stop qualification**:** 186.70

Function15 **(F15):**

$$\max\ f(x,y)=1+x\times\sin(4\pi x)-y\times\sin(4\pi y+\pi),\ \ x,y\in[-1,1]$$

Stop qualification**:** 2.2590

Function16 **(F16):**

$$\max f(x,y) = -(x^2+y^2)^{0.25}(\sin^2 50(x^2+y^2)^{0.1}+1.0),\ x,y\in[-1,1]$$

Stop qualification**:** -0.010

Function17 **(F17):**

$$\max f(x,y) = \frac{1}{2}(x^4-16x^2+5x+y^4-16y^2+5y),\ x,y\in[-5,5]$$

Stop qualification**:** 78.330

Function18 **(F18):**

$$\max f(x,y) = -(|x|+|y|+|x||y|),\ x,y\in[-10,10]$$

Stop qualification**:** -0.0010

Function19 **(F19):**

$$\max f(x,y) = -[\frac{1}{K}+\sum_{j=1}^{25}\frac{1}{c_j+\sum_{i=1}^{2}(x_i-a_{ij})^6}]^{-1},\ x,y\in[-65.536,65.536]$$

$$c_j = j;(a_{ij}) = \begin{bmatrix} -32,-16,0,16,32,-32,-16,0,16,32,-32,-16, \\ 0,16,32,-32,-16,0,16,32,-32,-16,0,16,32 \\ -32,-32,-32,-32,-32,-16,-16,-16,-16, \\ -16,16,16,16,16,16,32,32,32,32,32,0,0,0,0,0 \end{bmatrix};K\text{=500}$$

Stop qualification**:** -0.9990

Function20 **(F20):**

$$\max f(x,y) = -x\sin\sqrt{|y+1-x|}\cos\sqrt{|y+1+x|} + (y+1)\cos\sqrt{|y+1-x|}\sin\sqrt{|y+1+x|},\ x,y\in[-512,512]$$

Stop qualification**:** 511.70

Function21 **(F21):**

$$\max f(x,y) = -[(4-2.1x^2+x^{\frac{4}{3}})x^2+xy+(-4+4y^2)y^2],\ x,y\in[-5.12,5.12]$$

Stop qualification**:** 1112.145

Function22 **(F22):**

$$\max f(x,y) = \sin(x)\sin^{20}(\frac{x^2}{\pi})+\sin(y)\sin^{20}(\frac{2y^2}{\pi}),\ x,y\in[0,\pi]$$

Stop qualification**:** 1.80