A Self-adaptive Genetic Algorithm Based on the Principle of Searching for Things

Guoli Zhang
College of Mathematics and Physics
North China Electric Power University, Baoding, China
Zhangguoli@ncepu.edu.cn

Siyan Wang and Yang Li
College of Computer Science and Technology
North China Electric Power University, Baoding, China
wangsiyan1985@163.com; liyang20032006@126.com

***Abstract*—This paper proposes a new self-adaptive genetic algorithm。This new algorithm divides the whole evolution process into three stages. At each stage, the new algorithm adopts different operation method. The main ideas are grading balance selection, continuous crossover operation. The new algorithm designs especially self-adaptive mutation probability according to the principle of searching for things. Numerical experiments show that the new algorithm is more effective than the comparative algorithm in realizing the high convergence precision, reducing the convergence generation and good at keeping the stability of the adaptive genetic algorithm.**

***Index Terms*—adaptive genetic algorithm, gray code, simulated annealing, grading balance, continuous mutation.**

I. INTRODUCTION

The genetic algorithm (GA), proposed by Holland in 1975[1], is a kind of intelligent approaches, is now generating interest in the research community as an alternative for solving many problems because of its good characteristics such as simplicity, minimal problem restrictions, and global perspective.

Both the natural selection theory of Darwin and the genetic mutation theory of Mendel tell us that the evolution of living things is finished through reproduction, mutation, contest and selection. If the practical problem could be described as an optimization problem for an objective function, the procedure of GA will be interpreted as the fitness of the population to the environment and the variables as the individuals, and then starting from the current population, a new generation will be generated through appropriate reproduction, crossover, and mutation. The process will be repeated till the required population or the specified limit of time.

Standard genetic algorithms assess individuals by their chromosomes, it can separate them from other individuals in the evolutionary system[3]. Individuals receive a fitness score determined by a static, absolute fitness function. Individuals with the same chromosome receive the same fitness score, independent of the generation. Chromosome diversity can't be guaranteed, which results in the occurrence of premature convergence.

Adaptive genetic algorithm proposed by Srinvivas[4] is an improvement of the basic genetic algorithm. By using adaptive operator, the high convergence speed and the high convergence precision has been obtained. Adaptive genetic algorithm is not only keeping the stability of the algorithm but also reducing the convergence generation.

The paper proposes a new adaptive genetic algorithm. This new algorithm divides whole the evolution process into three stages. At each stage, the new algorithm adopts different operation method. The main ideas are grading balance selection, continuous crossover operation. The new algorithm designs especially self-adaptive mutation probability according to the principle of searching for things. The detail can be seen in section . In the process of searching the optimal solution, the new algorithm is not only keeping the diversity of the population but also preserving the good individuals. Numerical experiments show that: convergence speed, global convergence capability and the stability of the algorithm are all improved obviously.

II. REGION BALANCE VARIATION ALGORITHM

As a widespread phenomenon in the real world, optimization is essentially to choose the best one from finite or infinite options. The current optimization methods include differential, hill climbing, combinatorial options, simulated annealing, neural networks and genetic algorithms and so on. But no matter what kind of method is used, it is difficult to get an effective hunting way for the optimization problem when the varying region of the variable is large. Theoretically speaking, for a given

doi:10.4304/jcp.5.4.646-653

optimization problem, the accurate optimal solution should be found if it exists. However, for practical optimization problems, the satisfying solution is usually focused because of the existing of theoretical error of the model, the information error in the data and the cognition bias. Generally, it is difficult to find the optimization solution or satisfying solution for the problems with large varying region of variables. Accordingly, for optimization problems with large-scale and high-accuracy, it is obviously to help us to find the optimization solution or satisfying solution by reducing the hunting region gradually without losing the optimal solutions.

In this paper, the new algorithm is trying to adjust the genetic strategy in order to get the optional solution of the problems.

Initialize population and encoding

At first, we set some parameters of the new algorithm.

In this paper, the population impresses as $P = \{x_i | i = 1,2,\cdots,L\}$, L is the size of the population, each individual x_i is described as follows:

$$x_i = (x_i(1), x_i(2), \cdots, x_i(n)) \tag{1}$$

where n is the number of the decision-making variable.

The binary code method is a basic encoding strategy, but it is not easy to reflect the characteristic of the problem' structure, such as some optimization problems of the continuous function. In addition, the random character of the genetic algorithm makes its local search capabilities very poor. In order to make up for these deficiencies, the paper uses gray code to encode the individual. Gray code can effectively reflect the characteristic of the problem' structure and enhance calculating speed.

Gray code encodes as follows:

Assume that the original binary code of the individual is:

$$y_i \to [y_i(0), y_i(1), \cdots, y_i(m)] \tag{2}$$

then the gray code is :

$$y_i \to [c_i(0), c_i(1), \cdots, c_i(m)] \tag{3}$$

The encoding function is that $c = G(y)$, $\oplus$ is XOR, and m is the length of the code. Encoding method is as follows:

$$\begin{cases} c_i[k] = y_i[k] \oplus y_i[k+1] & (k \in N, 0 \le k \le m-1) \\ c_i[m] = y_i[m] & (k = m) \end{cases} \tag{4}$$

Calculate the fitness of each individual

The fitness' of individuals are an important factor in the genetic algorithm, and should be calculated by a proper method in order to enhance performance of GA.

Simulated annealing algorithm has been used in various combinatorial optimization problems and has been particularly successful in circuit design problems and so on. Simulated annealing thinking is used in the paper.

In this paper, the fitness function is based on the objective function values. It turns the value of the objective function into fitness with simulated annealing thinking and then carries out the scale transformation. The fitness scaling algorithm formulas are as follows:

$$T = T_0 \times 0.99^{gen} \tag{5}$$

$$sum = \sum_{i=1}^{L} \exp(\frac{f(x_i)}{T}),\ i = 1,2,\cdots,L \tag{6}$$

$$f_i = \frac{\exp(\frac{f(x_i)}{T})}{sum}, i = 1,2,\cdots,L \tag{7}$$

where T_0 is the initial annealing temperature which should be big enough; T is the temperature of the current generation; gen is the current generation, $gen = 1,2,\cdots,Mgen$, $Mgen$ is the maximum evolutionary generation; L is the size of the population; $f(x_i)$ is the objective function value of the individual x_i; f_i is the fitness of the individual x_i.

The fitness calculation strategy which is based on simulated annealing thinking can express the fitness of each individual properly. The new strategy is more effective than the basic genetic algorithm's fitness calculation method and some other fitness calculation method. It is good for our next work.

Genetic operator

This paper adopts grading balance selection, uses continuous crossover and continuous mutation strategy in each stage to get the crossover probability and the mutation probability.

First of all, divide the evolutional generation into three parts. If the maximum evolutional generation is $Mgen$, plot out as follows:

The first part: $[0, 0.382Mgen)$;

The second part: $[0.382Mgen, 0.618Mgen)$;

The third part: $[0.618Mgen, Mgen)$.

1) Grading Balance Selecting Strategy

In this section, we should do like this: first of all, calculate the fitness of each individual by formula(5)-(7), then sort the individual of the population in descending order of fitness. Then if the evolutional generation in the interval of $[0, 0.382Mgen)$, the selection method is stochastic tournament selection; if the evolutional generation in the interval of $[0.382Mgen, 0.618Mgen)$, we adopt the method as follows:

The parameter L is the size of the population. The select method is an enhancing method at both ends. This method can be described as follows: firstly, extracting $\lfloor L/4 \rfloor$ individuals from the population which own higher fitness, then calculate their new fitness by the

formula $\frac{f_1}{f_{\lfloor L/4 \rfloor}} f_i$; secondly, extracting $\lfloor L/4 \rfloor$ individuals from the population which owns lower fitness, then calculate their new fitness by the formula, $\frac{f_{\lfloor 3L/4 \rfloor}}{f_L} |\cos(\frac{f_i}{f_{\lfloor 3L/4 \rfloor}} \pi)|$ sorting the new fitness in descending order, and then selecting the individuals by roulette wheel selection. In above formulas, $\lfloor \bullet \rfloor$ represents maximum integer which is less than or equal $\bullet$. The third part is $[0.618Mgen, Mgen)$, in this stage, selection method is optimal preservation strategy.

The selection strategy of the first step drives down the probability of the best individual be eliminated in the initial stage of the evolution, maintains the diversity of the population. The selection strategy of the second step upgrades the selection probability of the individual on the edge of the population, at the same time, this method upgrade the elimination probability of the individual in the vicinity of the individual which has the average fitness. It ensures the diversity of the population as well as through the elimination of some of the individual which has little mean to improve the convergence value. The selection strategy of the third step accelerates the convergence rate, makes the algorithm getting to the optimal solution more quickly. This selection strategy has provided more opportunities to extract better individual for the crossover operation.

Numerical experiment shows that the co-selection strategy is effective in realizing the good individuals that can be used in the crossover operation.

2) Continuous crossover probability and continuous mutation probability

In GA, choosing appropriate crossover probability p_c and mutation probability p_m is the key factor to increase the performance and convergence of the algorithm. If P_c is over-large, the crossover will destroy the excellent scheme, which has bad influence on the evolutional process; while if P_c is over-small, the rate of generating the new chromosomes is slower. If the mutation probability P_m is over-larger, the more new chromosomes are generate, however it will destroy many better chromosomes, so the performance of the GA approximate to that of the random search; while if P_m is over-small, the capability of generating new chromosomes and restraining premature convergence is worse. Hence, it is important to choose appropriate P_c and P_m .

In the past, majority adaptive crossover probabilities and mutation probabilities are related to the iterative time. In the beginning, they kept the diversity of the population by rough search, in the end, in order to ensure the algorithm can reserve the optimal solution and improve the local search speed, they carried out searching carefully.

In this paper, a new method is adopted. This paper takes advantage of the principle of searching things in our daily life. When someone search something, if he or she can not find it in one place after a period of time , he or she maybe goes to another place and search it in this place. What is called turning to another place is that through a big mutation probability, obtaining some new individual from a new region that is far from the ancestor of them.

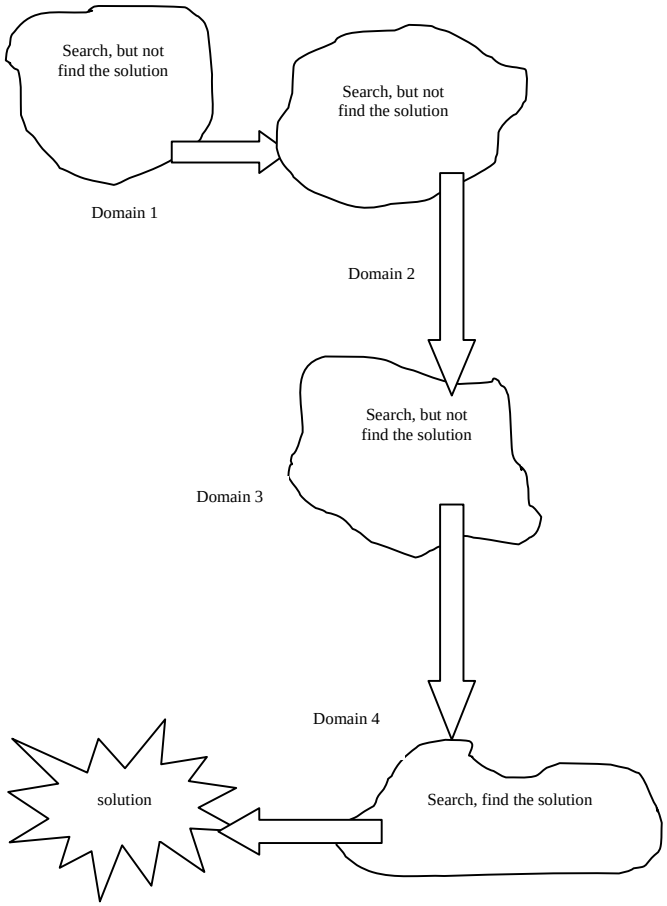

Figure 1. search method of the algorithm with continuous mutation

The calculating method of the crossover probability is different from the computation method of the mutation probability. When the crossover probability is changing from a small probability to a large probability, the individual is not need to go to another domain, it still in their own domains.

Let us set the parameters as follows:

p_c^0 is the initial crossover probability;

p_c^1 is the probability descending unit of the crossover probability;

p_c^2 is the lower limit of the crossover probability;

p_m^0 is the initial mutation probability;

p_m^1 is the probability descending unit of the mutation probability,;

p_m^2 is the lower limit of the mutation probability.

If the evolutional generation is in the interval $[0, 0.382Mgen)$, let p_c^0 =0.9, p_c^1 =0.05, p_c^2 =0.7;

If the evolutional generation is in the interval $[0.382Mgen, 0.618Mgen)$, let p_c^0 =0.7, p_c^1 =0.05, p_c^2 =0.5;

If the evolutional generation is in the interval $[0.618Mgen, Mgen)$, let p_c^0 =0.5, p_c^1 =0.05, p_c^2 =0.3;

If the evolutional generation is in the interval $[0, 0.382Mgen)$, let p_m^0 =0.1, p_m^1 =0.01, p_m^2 =0.05;

If the evolutional generation is in the interval $[0.382Mgen, 0.618Mgen)$, let p_m^0 =0.15, p_m^1 =0.01, p_m^2 =0.1;

If the evolutional generation is in the interval $[0.618Mgen, Mgen)$, let p_m^0 =0.3, p_m^1 =0.01, p_m^2 =0.15;

In each stage, we use different crossover probability and mutation probability. At the first stage, the crossover probability is larger than the crossover probability in other stages, thus, we can keep the diversity of the population. At the third stage, the crossover probability is smaller than the crossover probability other stage, thus, we can preserve the good individuals that generated from the process of the evolution. The mutation probability is not the same with the crossover probability. The mutation probability in front is smaller than the mutation probability in latter. In this way, at the end of the evolutionary stage, the algorithm can out of the local optimal solution, so the algorithm can quickly get to the global optimal solution.

There are two flow charts as follows: Figure 2 is the crossover probability's changing flow chart, Figure 3 is the mutation probability's changing flow chart.

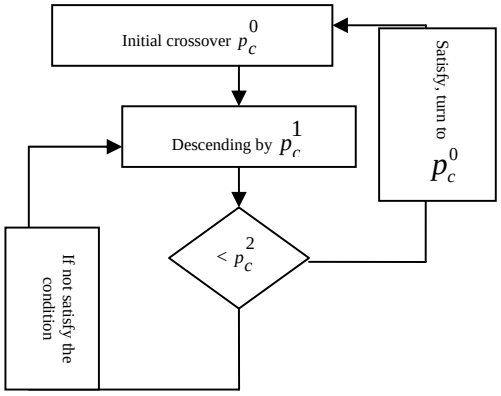

Figure 2. adaptive crossover probability flow chart

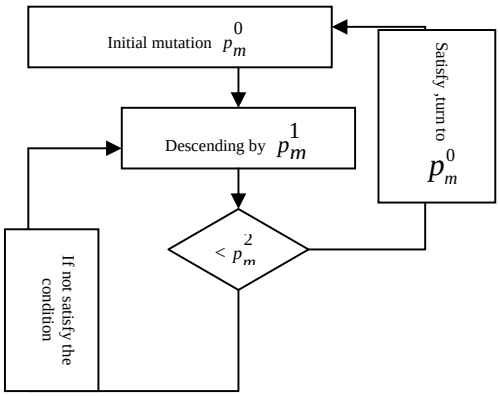

Figure 3. adaptive mutation probability flow chart

In the Figure 2, p_c^0 is the initial crossover probability, p_c^1 is the probability descending unit of the crossover probability, p_c^2 is the lower limit of the crossover probability; p_m^0 is the initial mutation probability, p_m^1 is the probability descending unit of the mutation probability, p_m^2 is the lower limit of the mutation probability.

From the flow chart we can know that: in the new algorithm, the mutation probability is various by descending order. In this way, mutation probability can vary continuously. It is a comprehensive and balanced strategy. When we use the new algorithm to test the benchmark functions, we have found that it is an effective method. Not only the convergence speed and convergence precision have improved, but also the stability of the algorithm performs well.

Moreover, the crossover operation as follows:

Let the two parents are:

$$c_1 = (c_1(0), c_1(1), \cdots, c_1(m)) \quad (8)$$

$$c_2 = (c_2(0), c_2(1), \cdots, c_2(m)) \quad (9)$$

firstly, create a crossing l between 1 and m randomly, and a random number b, if $b \le p_c$, a =1,if $b > p_c$, a =0; Suppose the new individual after crossover are $c_1^{'}, c_2^{'}$, then:

$$\begin{cases} c_1^{'}(j) = c_1(j) \\ c_2^{'}(j) = c_2(j) \end{cases} \quad (j = 1, 2, 3, \ldots, l) \tag{10}$$

$$\begin{cases} c_1^{'}(j) = a \times c_1(j) + (1-a) c_2(j) \\ c_2^{'}(j) = a \times c_2(j) + (1-a) c_1(j) \end{cases} \quad (j = l+1, \ldots, m) \tag{11}$$

3) The flow of the new algorithm

a) initialize running parameter;

b) choose an initial population, code by gray code;

c) calculate the fitness of each individual;

d) judge the end condition, if satisfies the end condition, end operating; if not, operate as follows:

Firstly, perform grading balance selecting strategy;
Secondly, perform continuous crossover;
Thirdly, perform continuous mutation;
Finally, obtain a new population.

e) turn to step (c).

III. NUMERICAL EXPERIMENTS

In this section, the testing functions are presented to illustrate the feasibility and efficiency of the adaptive genetic algorithm. In addition, the paper also uses an adaptive genetic algorithm that is similar to the new algorithm, only its genetic operator is different from the new algorithm. Through comparative trial, we can know that the new algorithm is an effective method.

In the following, three functions are tested. The goal of the test is to find the maximum of the function. The testing population size is 200, testing generation is 100, testing environment is visual c++6.0. The data of the experiment is average result. T_0 of AGAES and AGA is 10000, π 为 3.14159265358.

The first experiment tests the function with a variable and with some local optimal solutions. Through the test, we can prove that our new algorithm is an effective algorithm in realizing the global optimal solution in a short time and with a less convergence generation than other algorithms.

$$\begin{cases} f_1(x) = x\cos 2x + 2.0 \\ x \in [-\pi, \pi] \end{cases} \tag{P1}$$

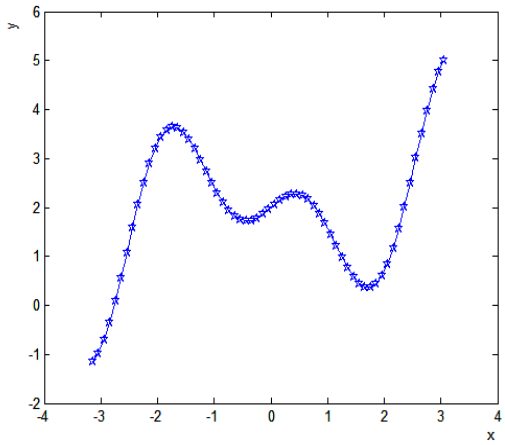

Figure 4. function image of P1

Figure 4 is the function image of P1. From the image we can see that there are some local optimal solutions. So the function can test the global search activity of the new algorithm. The comparative experiment as follows:

TABLE I. COMPARATIVE TRIAL OF AGAES AND AGA

test / method	Testing generation	Average convergence generation	Average result of f_1	Theoretic result of f_1
AGARBV	100	7	5.1415927	5.1415927
AGA	100	53	5.1415927	5.1415927

From TABLE I, we can see the new algorithm's average convergence generation is 7, but the AGA's convergence generation is 53. The new algorithm is quicker than the AGA in realizing the best solution of the objective function.

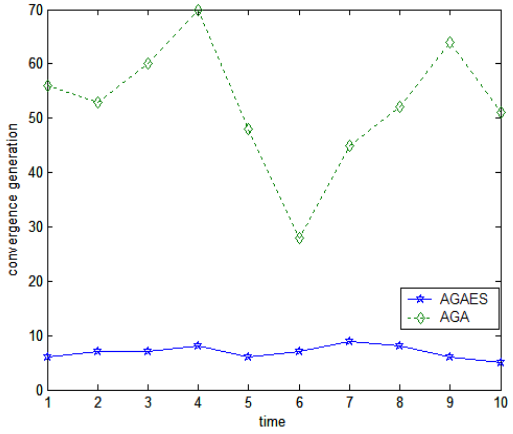

Figure 5. convergence generation of AGAES and AGA

From the figure 5, which gives the convergence generation of ten times, we can know that AGAES is more

stable than AGA, and the new algorithm can reach to the convergence value quicker than AGA.

When the variable x 's scope is in a large scope, the search speed and the convergence of the new algorithm is better than the comparable algorithms. In this experiment, we use the scope [-1000, 1000]. The detail experiment is as follows:

$$\begin{cases} f_1(x)=x\cos 2x+2.0 \\ x\in[-1000,1000] \end{cases} \quad \text{(P2)}$$

TABLE Ⅱ is the convergence comparative of the AGAES,AGA and SGA, figure 7 is the testing of the changing chart of the AGAES,AGA and SGA.

TABLE Ⅱ CONVERGENCE VALUE OF THE THREE ALGORITHMS

Method / Generation	AGAES	AGA	SGA
20	1001.03	996.283	57.457271
30	1001.03	996.283	86.479046
40	1001.03	996.283	116.909899
50	1001.03	997.141	138.795057
60	1001.03	998.468	156.052344
70	1001.03	999.133	183.739584
80	1001.03	1000.88	199.771360
90	1001.03	1001.02	218.759226
100	1001.03	1001.03	240.736910

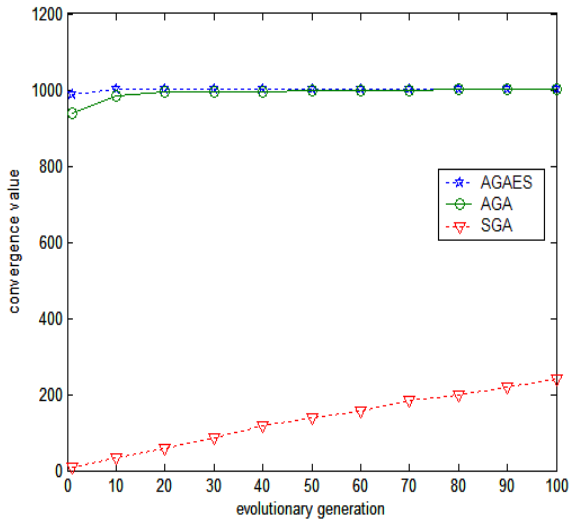

Figure 6. comparation of convergence value

From TABLE Ⅱ and figure 6, we can know that the new algorithm can quickly get to the optimal solution of the function when it is variable x in a large scope. Compared with the AGA, we can see that the new algorithm can escape the local optimal solution and reach to the global optimal solution with a high speed. So it is an effective algorithm in realizing the global optimal solution of the function when the variable of the function is in a large scope.

The second experiment test functions with two and three variables. In this section, we give a comparative table of three algorithms in the other paper. From the comparative data, we can see that the new algorithm is more effective than other algorithms. The two testing functions are typical benchmark functions.

$$\begin{cases} f_2(x)=-(x_1^2+2x_2^2-0.4\cos(3\pi x_1)-0.6\cos(4\pi x_2)) \\ x_1\in[-10,10], x_2\in[-10,10] \end{cases} \quad \text{(P3)}$$

$$\begin{cases} f_3(x)=(-4+2.1x_1^2-\frac{x_1^4}{3})x_1^2-x_1x_2+(4-4x_2^2)x_2^2 \\ x_1\in[-10,10], x_2\in[-10,10] \end{cases} \quad \text{(P4)}$$

TABLE Ⅲ COMPARATIVE TRIAL OF THE FOUR ALGORITHMS

Algorithm name	P % / N	Function P(3)	Function P(4)
SGA	P %	43	30
	N	124	152
AFGA	P %	83	77
	N	93	108
ADGAA	P %	97	93
	N	82	87
AGAES	P %	100	100
	N	10.7	12.9

P is the convergence probability of the function with the four algorithms. N is the convergence generation of the function with the four algorithms. The algorithm name is the algorithm that was proposed in the reference [21].

From TABLEⅢ, we can see that the new algorithm is an effective algorithm for the benchmark function. Function P (3) has some local optimal solution. But the new algorithm can find the global optimal value 1 in a short time with a high convergence probability and a precision result. Function P (4) also reaches to the global maximum value: 1.031628. Compared with SGA, AFGA and ADGAA, it can quickly reach to the convergence value.

$$\begin{cases} f_4(x)=\sqrt{x_1}+\sqrt{x_2}+\sqrt{x_3} \\ x_i \in [0,100](i=1,2,3) \end{cases} \quad (P5)$$

$$\begin{cases} f_4(x)=\sqrt{x_1}+\sqrt{x_2}+\sqrt{x_3} \\ x_i \in [0,10000](i=1,2,3) \end{cases} \quad (P6)$$

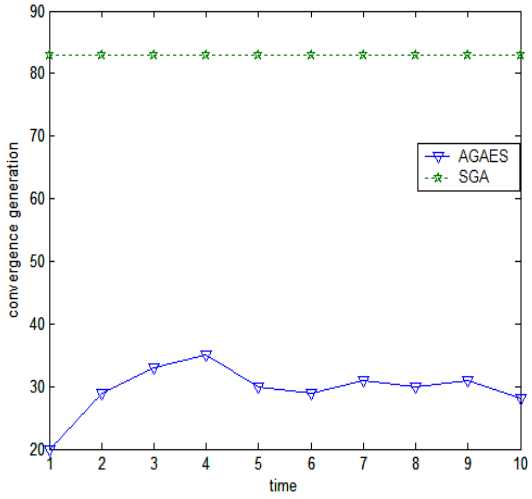

Figure 7. convergence generation of P5

TABLE Ⅳ COMPARATIVE OF SGA AND AGAES

Algorithm	Average convergence generation	Optimal solution	Theoretic convergence value
SGA	No convergence	52.586896	300
AGAES	29	300	300

The testing of (P5) and (P6) show us that when there are more than one variable, the algorithm is also an effective method to obtain to the global optimal solution. From the TABLE IV we know that the SGA can not get to the convergence value 300 within the testing generation. But the new algorithm can get to the convergence value in a short time. So it is not only a good method for functions with single variable but also an effective method for functions with many variables.

The third experiment test function and its image is as follows:

$$\begin{cases} f_5(x,y)=0.5-\dfrac{\sin^2\sqrt{x^2+y^2}-0.5}{\left(1+0.001(x^2+y^2)\right)} \\ -10 \le x \le 10, -10 \le y \le 10 \end{cases} \quad (P7)$$

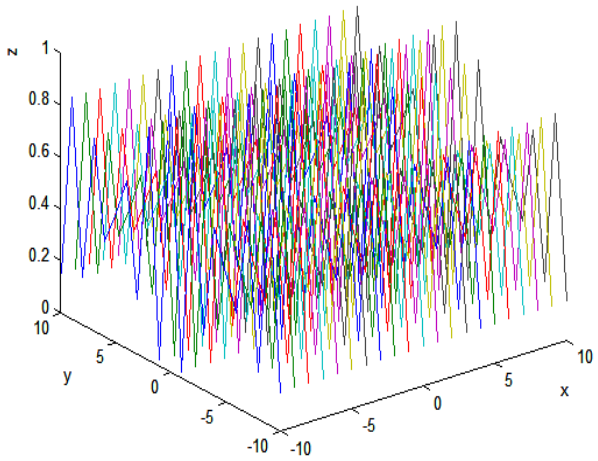

Figure 8. function image of P7

This function has many local optimal solutions, through the new algorithm, we can find the global optimal solutions in a few seconds with an average convergence generation of 42. Its convergence value is 1, and its convergence point is (0.0).

From the three experiments, we can see that the new algorithm can quickly reach to the global optimal solution whenever there are one or more variable. Compared with the other algorithms, such as AGA, AFGA, ADGAA, the new algorithm has a high convergence probability and a small convergence generation. So from the experiments, we can say the new algorithm that proposed in this paper is an effective one.

IV. CONCLUSION

In this paper, the new adaptive genetic algorithm which is based on evolutionary stages is an effective strategy. Through the improvement of the selection strategy and the using of the continuous crossover and continuous mutation method in every stage of the algorithm, the new method is considered to be a simple but a good method in the field of the self-adaptive genetic algorithm. The combination of the evolutionary stages strategy and the new idea that adopted by the selection, crossover and mutation stage, make a good result. The new algorithm can get to the global optimal solution with a high speed and with a few convergence generations. It can be used in image segmentation, knapsack problem, TSP problem and so on. It is a breakthrough of the genetic algorithm.

ACKNOWLEDGEMENT

This work is supported by the homecoming Fund (20514001) of North China Electric Power University.

REFERENCES

[1] J. H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. Michigan Press, 1975.

[2] F. Li, L. Liu and C. Jin. A Kind of Composite Genetic Algorithm Based on Extreme Pre-Judgement. Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design.2007pp:1038-1043

[3] B. Li, T. S. Lin, L. Liao, and C. Fan, Genetic Algorithm Based on Multipopulation Competitive Coevolution. 2008IEEE Congress on Evolutionary Computation.2008,pp:225-228

[4] S. M，and P. L. M, Adaptive probabilities of crossover and mutation in genetic algorithms, System, Man and Cybernetics, Vol. 24, No. 4, 1994,pp:656-667.

[5] D. Yan, J. Zhang, B. Yu, C. Luo and S. Zhang. A Genetic Algorithm for Finding Minimal Multi-homogeneous Bezout Number. Seventh IEEE/ACIS International Conference on Computer and Information Science. 2008,pp:301-305..

[6] J. Zhang, C. Liang, Q. Lu. A Novel Small-Population Genetic Algorithm based on Adaptive Mutation and Population Entropy Sampling. Proceedings of the 7th World Congress on Intelligent Control and Automation 2008,pp:8738-8742.

[7] X. L. Hu, Y. Tang, J. Cai. Optimal Approximation of Head-Related Transfor Function Based on Adaptive Genetic Algorithm. IEEE Int. Conference Neural Networks & Signal Processing .2008,pp:129-132.

[8] J. Li, C. Wang, Y. X. Yang. An adaptive Genetic algorithm and Its Application in Bilateral Multi-issue Negotiation. The Journal of China Universities of Posts and Telecommunications.2008,pp:94-97.

[9] R. Sinha, K. Swearingen. Comparing Recommendations Made by Online Systems and Friends. Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries,Dublin, Ireland,2001.

[10] K. S. Leung,Q. H. Duan, Z. B. Xu and C. K. Wang. A New Model of Simulated Evolutionary Computation-Convergence Analysis and Specifications, IEEE Trans, evolutionary computation,vol5,2001,pp:3-16

[11] G. M. Wang, X. J. Wang, Z. P. Wan, and S. H. Jia., An adaptive algorithm for solving bilevel linear programming problem, Applied Mathematics and Mechanics, Vol. 28, No. 12, 2007,pp:1605-1612.

[12] M. Y. Li, Z. X. Cai, and G. Y. Sun, An adaptive genetic algorithm with diversity-guided mutation and its global convergence property , Journal of Central South University of Technology, Vol. 11, No. 3, 2004,pp:323-327.

[13] X. Zeng, T. Ma, Y. W. Lin, and Z. Y. Feng, Genetic algorithm for autonomic joint radio resource management in end-to-end reconfigurable systems, The Journal of China Universities of Posts And Telecommunications, Vol. 15, No. 1, March 2008,pp:11-17.

[14] L. Y. Jia, and C. H. Zhou, Strategy for Genetic Algorithms to Solve TSP Problem Quickly, Computer Engineering, Vol. 34, No. 5, 2008,pp:174-181.

[15] Q. Li, W. Zhang, Y. X. Yin, and Z. L. Wang, A new adaptive algorithm for regulating the probabilities of crossover and mutation, Vol. 23, No. 1, 2008,pp:79-83.

[16] Y. Sheng, C. Zeng, and C. F. Zhang, Adaptive Compact Genetic Algorithm and Simulation, Journal of System Simulation, Vol. 20, No. 5, 2008,pp:1167-1169.

[17] J. Wu, P. Li, and J. Zheng, Adaptive muti-parent genetic algorithm and its performance analysis, Systems Engineering and Electronics, Vol. 29, No. 8, 2007,pp:1381-1384.

[18] M. Chen, and S. Liu. An improved adaptive genetic algorithm and its application in function optimization, Journal of Harbin Engineering University, Vol. 28, No. 8, 2007,pp:875-879.

[19] H. J. C. Barbosa, A. C. C. Lemonge, C. C. H. Borges. A Genetic Algorithm Encoding for Cardinality Constraints and Automatic Variable Linking in Structural Optimization. Engineering Structures.vol(30),issue(12),2008,pp:3708-3723

[20] X. Yuan, L. Cao, L. Xia. Adaptive genetic algorithm with the criterion of premature convergence.Journal of Southeast University, Vol. 19, No. 1 ,2003,pp:40-43.

[21] L. Cai. Real-coded adaptive genetic annealing algorithm based on distance measurement. Journal of Shenzhen University Science and Engineering, vol.21, No.4, Oct. 2004,pp:291-294.