Test Case Generation Based on Hierarchical Genetic Algorithm

Liu Shurong
School of Software
Beijing Institute of Technology
Beijing, China
brendaliu1989@foxmail.com

Hu Changzhen
School of Software
Beijing Institute of Technology
Beijing, China
chzhoo@bit.edu.cn

Xue Jingfeng
School of Software
Beijing Institute of Technology
Beijing, China
xuejf@bit.edu.cn

Li Zhiqiang*
School of Software
Beijing Institute of Technology
Beijing, China
*Corresponding author: lizq@bit.edu.cn

Abstract—The basic genetic algorithm was proposed to optimize the test case generation. It has been applied widely. Based on basic genetic algorithm, this paper proposed the hierarchical genetic algorithm to generate test cases. The hierarchical genetic algorithm divided the initial population into hierarchical son population and operated selection, crossover and mutation among son population independently. In the hierarchical genetic algorithm, the evolution of population was firstly operated between all layers, if the algorithm can't get the best test cases, it entered the next generation. Using this mechanism, the hierarchical genetic algorithm can avoid effectively 'inbreeding', 'local convergence', 'slow convergence' phenomenon. So it was the better way to generate test cases. This paper did the experiment using 3 benchmark program: triangle classification, bubble sort, the Max and Min. The experimental results show that the quality of test cases and the efficiency of generating test cases are improved markedly by hierarchical genetic algorithm compared with the basic genetic algorithm.

Keywords-hierarchical genetic algorithm; test case; son population; local convergence; benchmark program

I. INTRODUCTION

With the continuous development of software technology and software scale, software testing is becoming increasingly important role. In software testing, the selection of test case is a problem for structural testing. The quality of test case determines whether the code error can be detected as expected. With the development of automated testing tools, at present there are a number of methods for generating test case automatically, such as finite state machine (FSM), genetic algorithm (GA) and so on [1]. This paper aimed to improve the basic genetic algorithm to optimize the test case generation.

Genetic algorithm referenced biological mechanisms of natural selection and evolution and developed to randomize and adaptive searching algorithm [2]. In the 1960s, John Holland proposed genetic algorithm firstly, it is used to solve optimization problems of the searching algorithm. Genetic algorithm maps the inputting data domain D to gene domain G by coding technology and determines the searching method by the generic operator selection, crossover, mutation and survival of the fittest. Because it uses populations organized search, so you can simultaneously search multiple areas within G [3]. The operator crossover and mutation can generate new entity in populations. So it's more conducive to find the global optimal solution and avoid the local optimal solution. In the 1960s, after the genetic algorithm was proposed, it was applied in several ways.

Although the genetic algorithm has good global searching capability, but in the last stage of searching genetic algorithm prone to "local convergence", "slow convergence", such that the optimal solution is not a global optimum. For these problems, some people solve it by optimizing the fitness function and some people solve it by optimizing the genetic operator [4]. This paper solves it by Hierarchical genetic algorithm. Hierarchical genetic algorithm divides initial population into sub-populations to form a hierarchical tree structure, then operates the corresponding genetic operations on each layer of the tree. That takes full advantage of the excellent information between different layers and also avoids "inbreeding "phenomenon.

II. RESEARCH

Using genetic algorithm to generate test data still faces many shortcomings, such as 'local convergence', 'premature question', 'inbreeding' phenomenon and so on. For different problems, people have proposed different solutions. These solutions are mainly divided into three areas: improvement of the fitness function, improvement of the genetic operator, the improvement of initial population.

Literature [5] mainly improved the initial population and proposed the genetic particle swarm algorithm. Its main idea was that the entire initial population was divided into t populations, and then chosen individual whose fitness value was minimum to cross-breeding between different groups. This way of crossing between populations actually controlled the Hamming distance of crossing individual, avoided inbreeding and helps maintain the diversity of population. In order to improve the convergence speed, in the same group individual was randomly chosen to cross-breeding. Before mutation this algorithm used elitist strategy to retain good individuals.

Literature [6] mainly improved the fitness function and genetic operators. It used the evaluation function from the constraints on the way to generate test data. That avoided the blindness of the searching data process and has the higher the efficiency and was unnecessary to care the problem that was matching of test data and the test sequence generation. Even there was only the evaluation function to provide an evaluation value to the point of the searching space, but also it can operate efficiently, which makes it become a powerful searching algorithms.

In Literature [7], Bui T U and Moon B R proposed the Hierarchical genetic algorithm(HGA). They initially proposed hierarchical genetic algorithm to solve the problem of segmentation map. Hierarchical genetic algorithm effectively solved the problems "local convergence" and "inbreeding", and is applied in different fields. In [8] and [9] the hierarchical genetic algorithm were applied in the network to optimize task scheduling and power transformer design, greatly improving the convergence rate.

III. HIERARCHICAL GENETIC ALGORITHM

A. *The Base Model of HGA*

Base genetic algorithm(BGA) changed all optimization variables to chromosomes by encoding. These chromosomes formed population and operated evolution. Hierarchical genetic algorithm divided initial entire population into hierarchical sub-population. For example, in Figure 1 there are three layers, and each layer has two sub-populations, you can see the details by Fig.1:

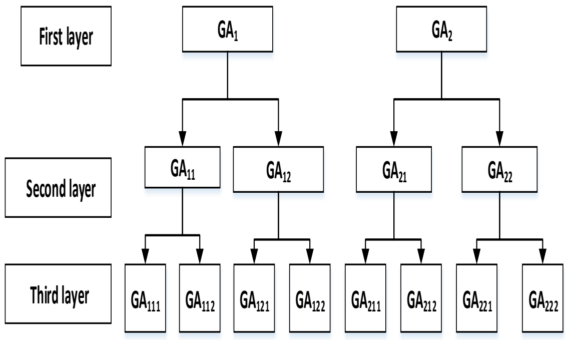

Figure 1. The population distribution of HGA

In Fig.1 there are two sub-populations in the first layer, each sub-population in the first layer has two sub-populations in the second layer, in total there are 2*2 sub-populations in the second layer. Each sub-population in the second layer has two sub-populations in the third layer, in total there are 2*2*2 sub-populations in the third layer. So, if there is K1 sub-populations in the first layer and there is K_n sub-populations in the (n-1)*th* layer. The number of the sub-populations in the n*th* layer [10] is :

$$\prod_{i=1}^{n} K_i \qquad (1)$$

The essence of the Hierarchical genetic algorithm is that the upper layer operates lower layer as sub-population. We took 3-layer hierarchical genetic algorithm as example and figure 2 is its flow char. In Fig.2 the first step is the initial population stratified according to Fig.1, at the same time, the genetic variable t=0 and define hierarchies counter m = 3 for circulating layers. Firstly, the hierarchical genetic algorithm operated selection, mutation, crossover in the same layer, if we can't get the best test case for the target path, then m-1 and enter the next layer to searching the best test case for target path and so on until you find the best test case for your target path.

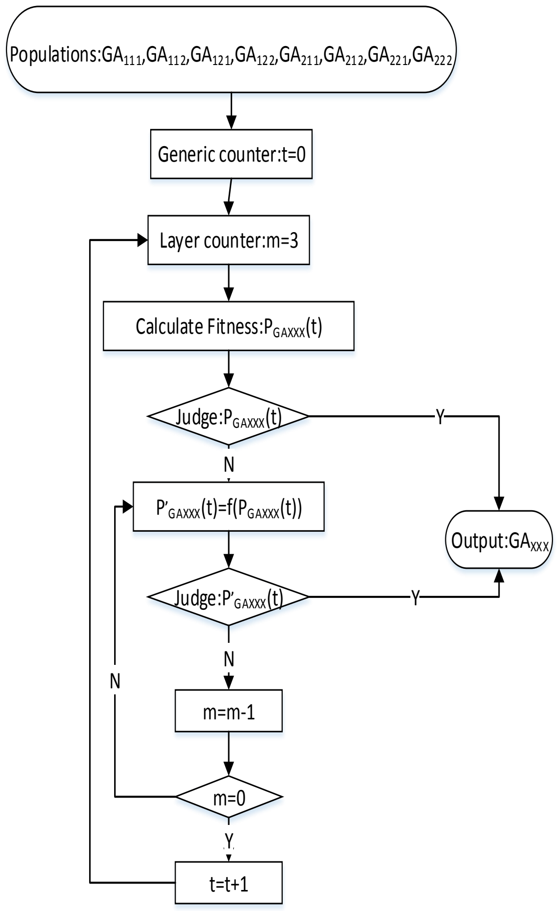

Figure 2. The flow chart of HGA

B. *The Fitness Function of HGA*

According to specific hierarchical genetic algorithm to generate test cases, we choose the fitness function based on the Huffman coding method, which codes all paths by the Huffman coding, then cartridge the corresponding program. Finally, get the fitness function value of the corresponding test cases by matching method proposed by Ahmed.

We signed the Tth generation of evolutionary population as x, x is the input data of the program testing. Sign the path string as f(x), the length of f(x) is |f(x)|, in total n testing path, $f_{j(j=1,2,\ldots,n)}$ is the jth testing path. We can get the fitness function P(x) by calculating the matching value of f(x) and f_j.

The specific calculation formula is as follows:

$$P_j(x) = (C_j(f(x))+1) * P_j^{'}(x) \tag{2}$$

$$P_j^{'}(x) = \sum_{k=1}^{\min(|f(x)|,|f_j|)} m_{jk} * d_{jk}(f(x)) \tag{3}$$

m_j-Sign the number of the same bit of the f(x)and f_j;
d_{jk}-judge whether the kth bit of f(x) and f_j is the same, iftrue,d_{jk}=1;if false d_{jk}=0;
c_j(f(x))-the number of the continuous same bit starting from first bit;
m_{jk}-the number of same bit of f(x) and f_j when finished the kth bit;

we can get the matching degree of f(x) and fj by calculating Pj(x).

C. *The Genetic Operator of HGA*

a) *The Encoding Method of HGA*

The base genetic algorithm has many coding methods, such as binary coding, real coding, gray coding, sign coding, natural coding. According to the hierarchical genetic algorithm used to generate the test case, we use binary coding that is used commonly in base genetic algorithm. Binary coding coded the individuals of the initial population to a binary string by binary symbols '0' and '1'.Using binary coding has many advantages, it makes the decoding and encoding more simple; At the same time, it makes the genetic operator selection, crossover, mutation more convenient.

b) *The Selection Operator of HGA*

We use simple and effective roulette wheel selection as selection operator and set selection probability Pc=10%, that said that the 10% individuals of the populations were selected and the 10% individuals of the populations were eliminated, in order to maintain the population size.

c) *The Crossover Operator of HGA*

The performance of the new populations generated by selection is improved, but it doesn't generate new chromosomes. In order to generate new chromosomes, genetic algorithm modeled on the way hybrid in biology, cross transposition on some part of the chromosome. First, to determine the probability Pi of crossover, approximately 0.8 ~ 0.95, which is about 80% ~ 95% individuals to perform cross, and then using the roulette wheel selection method, according to the fitness value selection is cross individual, using the method of random pairing of two individual selection, using the method of random positioning decided to cross position.

d) *The Mutation Operator of HGA*

The selection of chromosome variation and the determination of mutation position are all generated with the method of random. It's the first to determine the mutation probability Pm, and in order to prevent too many mutations result in slow convergence speed, the Pm is usually small, is about 0.001 ~0.01, which is to say there was 1~10 of 1000 characters mutations.

IV. EXPERIMENT AND APPROACH VALIDATION

We use three benchmarks for experimental verification of the method. These three benchmarks are triangular classification, bubble sort, maximum minimum. The structure of these three benchmarks are varies. Depending to TABLE I, for each set of experiment, we used a hierarchical genetic algorithm and genetic algorithm 500 times for each run under the same experimental environment, the initial population and the same parameter settings. The experimental procedures are written in java, running in the eclipse environment, the experimental parameters Settings as shown in TABLE II.

TABLE I. THE INTRODUCTION OF PROGRAM

Program Name	**Number of Branches**	**Composition of Branches**
Triangle Classific ation	3 select	3 layer selection nested
Bubble Sort	2 loop,1 select	2 layer loop nested
Max and Min	1 loop, 2 select	Loop nesting 2 parallel selection

TABLE II. THE INTRODUCTION OF PROGRAM

Para meter	**Selectio n mode**	**Crossover mode**	**Crossover robability**	**Mutation mode**	**Mutation probability**	**encoding**
Value	roulette wheel	Single poir	0.1	Single point	0.4	Binary

V. EXPERIMENTAL RESULT

A. *The Experiment of Triangular Classification*

Range of different data classification procedures for the triangle, the target path select one of the four paths, the path to find the target set for the termination condition. Laboratory records every time to find the evolution of the target path algebra and the average of the evolution algebra for 500 times. Due to the evolution of each run time only a few milliseconds, so only recorded the total evolutionary time for 500 times. As the results listed in table 3, the algebraic and time ratio got by hierarchical genetic algorithm/basic genetic algorithms.

TABLE III. TEST RESULT OF TRIANGLE CLASSIFICATION

Data Scope	Size of Population	The Average Evolution Generation			Evolutionary Time （s）		
		HAG	*BGA*	*Ratio （%）*	*HGA*	*BGA*	*Ratio （%）*
$[0,100]^3$	40	25.2	64.3	39.2	0.16	0.33	46.4
$[0,200]^3$	60	38.6	75.4	51.2	0.17	0.42	39.5
$[0,500]^3$	80	46.5	334.2	13.9	0.64	3.22	20.0
$[0,1000]^3$	150	70.9	514.7	13.8	1.23	6.89	17.8
$[0,2000]^3$	300	78.7	635.9	12.3	2.45	20.7	11.9

The experimental results showed that: (1) With the increasing of the input data range, two methods to find the evolution of the target path test cases generation and evolutionary time increases, this is due to the increase of the input data range makes search harder. (2) For the different input data range, two methods of algebraic ratio and time ratio is different, the biggest difference between algebraic ratio and time ratio is 51.2% and 46.4%. That Indicated that this method requires only basic genetic up about half of the evolutionary algorithm method algebra and evolutionary time to find the data coverage of the target. (3) With the increase of the input data range, the time ratio is more and more small. Which show that the superiority of the method under the large range of input data is more obvious.

B. The Experiment of Other Two Programs

To further verify the performance of the hierarchical ge netic algorithm, the reference to two different structures of the other program, select the input data [0, 1000] for the c ase of three experiments.

When comparing method, the population size of the thr ee test procedures selected as shown in Table IV, the expe riments to find the rules of the target path test data as the t ermination conditions.record the evolution algebra and the general evolutionary time, the results listed in TABLE IV.

TABLE IV. TEST RESULT OF OTHER TWO PROGRAMS

Program	Size of Popu-lation	The average evolution generation			Evolutionary time （s）		
		HGA	*BGA*	*Ratio （%）*	*HGA*	*BGA*	*Ratio （%）*
Max and Min	200	58.3	238.4	24.5	1.245	6.34	19.6
Bubble Sort	100	24.6	39.4	62.4	0.23	0.45	51.1

As we can see from the table 4, for all the test program, layered algebraic average evolution of genetic algorithm a nd evolutionary world is less than the basic genetic algorit hm, the evolution algebra and evolutionary time scale big gest, hierarchical genetic algorithm is 62.4% and 51.1% of the basic genetic algorithm, which shows that to find the target path test data, the evolution of the hierarchical gene tic algorithm and evolutionary time algebra is about half o f the basic genetic algorithm, the results are in conformity with triangle classification procedures. This fully shows th at the three benchmarks, hierarchical genetic algorithm is superior to the basic genetic algorithm.

VI. CONCLUSION

It had been an important part in software testing field to findeffective method for generating test cases. This pap er on the basis of the basic genetic algorithm is proposed f or the initial population stratification on genetic algorithm to improve the convergence rate. At the same time, based on the classification of triangle program to verify the supe riority of hierarchical genetic algorithm, hierarchical genet ic algorithm in terms of evolutionary time evolution algeb ra has obvious advantages, the application of genetic algor ithms layered can greatly improve the software testing effi ciency.

By coding method in a longer path and a large target pa th, subject to different programming languages limit the le ngth of the string, a string cannot be stored all target path coding, at this point, how to store the target path is furthe r study. In addition, for different testing procedures are aut omatically generated path coding and automatic stub proc edures to improve the automation of software testing, whi ch need to be studied further too.

ACKNOWLEDGMENT

This work was supported by the Key Project of National Defense Basic Research Program of China (Grant No. B1120132031).

REFERENCES

[1] M. J. D. Powell, “The BOBYQA algorithm for bound constrained optimization without derivatives”, Technical report,Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, UK, 2009.

[2] M. Nehrir , C. Wang , K. Strunz , H. Aki , R. Ramakumar , J. Bing , Z. Miao and Z. Salameh. “A review of hybridzrenewable/ alternative energy systems for electric power generation: Configurations, control, and applications”. IEEE Trans. Sustainable Energy, vol. 2, no. 4, pp. 392-403,2011.

[3] D. E. Goldberg. “Genetic algorithms in search, optimization and machine learning”. MA:Addison-wesley Publishing Company,1989.

[4] T. N. Bui, B. R. Moon. “Genetic algorithm and graph partitioning”. IEEE Trans on Computer, 1996, 45(7): 841-855.

[5] X. P. Luo, W. Wei. “General discussion on convergence of immune genetic algorithm”. Journal of Zhejiang University (EngineeringScience), December 2005, 39(12): 2006-2011.

[6] J. C. Lin, P. L. Yeh. “Using Genetic Algorithms for Test Case Generation in Path Testing”. IEEE Transactions on software Engineer, 2000.

[7] V. Rajappa, A. Biradar, S. Panda. “Efficient Software Test Case Generation Using Genetic Algorithm Based Graph Theory”. First International Conference on Emerging Trends in Engineering and Technology, 2008.

[8] X. B. Tan, L. X. Cheng, X. M. Xu. “Test Data Generation Using Annealing Immune Genetic Algorithm”. Fifth International Joint Conference on INC, IMS and IDC.

[9] M. A. Ahamed, I. Herma. “GA-based multiple paths test data generator”. Computer & Operations Reaearch, 2008, 35(10): 3107-3124.

[10] B. F. Jones, D. E. Eyres, H. Sthamer. “A strategy for using genetic algorithms to automate branch and fault-based testing”. The Computer Journal, 1998, 41(2): 98-107.