2009 Fifth International Joint Conference on INC, IMS and IDC

Test Data Generation Using Annealing Immune Genetic Algorithm

X. B. Tan
The Institute of Computer, Foshan Vocational and Technical College,Guangzhou 510090, China
E-mail: xianbotan@yahoo.com.cn

Cheng Longxin
Institute of Policy and Management, Chinese Academy of Science
Chinese Academy of Science, CAS
Beijing 100080, China
E-mail: chenglx@mail.ustc.edu.cn

Xu Xiumei
Computer and Information Technology, Beijing Jiaotong University
Beijing Jiaotong University, BJU
Beijing 100044 China
E-mail: xuxiumeizi@sina.com

***Abstract*—With the development of software technology and the expansion of software project scale, software testing appears to be more crucial. And test data selection is one of the nodi during software structure testing because the suitability of test data may directly affect error detection. Notwithstanding existence of several methods to generate test data automatically, such an algorithm overcoming disadvantages of the existing methods in practice hasn't been brought out, that some errors still have to be detected by engineering experience. Therefore, this paper analyzes the characteristics and shortcomings of simple genetic algorithm, simulated annealing genetic algorithm as well as immune algorithm respectively. Aiming at solving the shortcomings in standard Genetic Algorithm on search efficiency, individual diversity and premature, the Annealing Immune Genetic Algorithm (AIGA) is presented as the core algorithm of test data generation by introducing the mechanism of reproduction rate adjustment of individual concentration of immune algorithm and annealing principium into genetic algorithm. Finally, AIGA mentioned above was applied and verified with a practical software testing example.**

***Keywords*- Software testing; test data generation; genetic algorithm; expectation of reproduction.**

I. Introduction to Software Testing Technology

Software testing technology is an important way for ensuring the quality of software, before which, test data should be obtained at first. At present, forward verification and backtrack are generally adopted by testing personnel to generate test data manually for assigned path with disadvantages of high workload and low efficiency and quality of test data. Automatic test data generation can effectively generate test data for software testing. Consequently, how to generate test data automatically with high efficiency and accuracy has become a focus of the research in this area.

At present, the study of automatic test data generation is mainly concentrated in two aspects: test data generation based on program specification and generation based on program structure. Chuangai Sun et al (2001) [1] presented an automatic test data generation method based on Boolean specification. Lv Gang (2003) [2] presented a test data automatic generation method based on the specification of relational algebra query expression. These two methods are both black-box testing technology. Random testing data generation, generation based on symbolic execution or program execution is all automatic test data generation method based on program structure. D. Bird (1983)[3] presented the technique of automatically generating random software test cases, which could generate a large quantity of test data randomly without limitation of the number but the time to execute the program if too long because of the huge number of testing data. Nowadays, many testing tools adopted this method. P.D. Coward et al (2007)[4] presented a methodology for testing data generation based on symbolic execution, the main idea of which is that it executes the program of designated route W to get the values of variables, using the values of symbols as its input. Korel (1990) [5] using stepping method to execute the program, that is only one branch predicate is executed in one step. Korel also gave a definition to branch function, measuring the degree of current solution to optimum solution. Beside the technique mentioned above, Program Instrumentation and iterative relaxation method are both belong to dynamic test data generators.

In recent years, artificial intelligent technology such as genetic algorithm and so on was used in software testing, which has become a new research focus. As to this, a methodology for automatic test data generation based on Annealing Immune Genetic Algorithm was investigated in this paper, aiming to develop an algorithm which could overcome the disadvantages of current test data generators after analyzing the characters of existing methodologies.

II. Analysis of common test data generators and search algorithm

A. Analysis of test data search algorithm

1) Genetic Algorithm: Genetic algorithm is a kind of compute simulation search technique by simulating the natural adaptation process of genetic selection and natural elimination, which was firstly presented by J.Holland [7] in 1975. As a quick, simple global search heuristic with good fault tolerance, genetic algorithm, which is widely used in pattern recognition, neural network, image processing and such areas, has an obvious advantage in the optimization process of kinds of structured objectives. Genetic algorithm can only find an approximate solution but not the global optimal solution in short time, and it can't guarantee converging to the global optimal solution.

2) Simulated Annealing algorithm: Simulated annealing algorithm, simulating the theory of solid annealing, is an extension of local search algorithm, which chooses the neighbor of maximum-cost state with a definite probability. The algorithm uses the mechanism of solid annealing to simulate combinatorial optimization problem, with internal energy E as the value of objective function , temperature T as control parameter . The mechanism of annealing algorithm is that begin with initial solution and initial value of , repeating the operation of "generating new solution, computing the value of objective function, accepting of abonding" with the decreasing of temperature and the current solution is the solution to the problem when the ending of algorithm executing. Simulated annealing algorithm, with wide application, is a generic probabilistic meta-heuristic algorithm based on Monte Carlo iterative solution approach. But it is difficult to control the parameters such as the initial value of temperature T, annealing velocity, which should be adjusted several times dependent on the results of the experiment.

3) Immune genetic algorithm: Immune genetic algorithm is a kind of improved genetic algorithm which systematically combines the Genetic Algorithm with the immune mechanism of creature to simulate the process of creatures' immune system for solving the optimum problem of engineering application. It considers invasive antigen as the objective function, and the antibody as solutions of the problem, using the affinity of antigen and antibody to denote the approximation ratio between the feasible solution and the optimal solution to the problem.

4) Proposal of the Annealing Immune Genetic Algorithm: Simply put, notwithstanding the practicality and stability of genetic algorithm in solving optimum problem, in practical application it possibly encounters problems of premature convergence, low local search ability, and low search efficiency in the middle and later period, while simulated annealing algorithm has the advantage of good local search ability, so introducing the mechanism of annealing algorithm to genetic algorithm to form a hybrid genetic algorithm is an effective way to promote the solution efficiency and the quality of outputs, which can be regarded as a new direction for further improvement on search ability of genetic algorithm. The Annealing Immune Genetic Algorithm presented in this article has two obvious features as follows:

(1)AIGA uses expected-breeding ratio, replacing past fitness to determine the individual reproduction opportunity, aiming to maintain individual diversity to make the algorithm more efficiency.

(2)Annealing temperature T, introduced from simulated annealing algorithm, was used to regulate the expected-breeding ratio to make the algorithm evolving to the direction of higher fitness much more accurately with the mechanism as that: at the beginning, the initial value of annealing temperature is largest, so the concentration of antibody had obvious impaction on expected-breeding ratio that various antibodies were selected to be cloned, as the algorithm processing, annealing temperature T was descending continually, and the impaction of antibody concentration to expected-breeding ratio was descending until to having no impaction. So at this case, the selection of antibody depended on concentration of antibody and annealing temperature initially, fully determined by the affinity of antigen that the antibody with higher affinity to antigen was selected to be cloned and the concentration obtained its maximum value.

B. The Automatic Test Data Generation Based on AIGA

1) The process of test data generating: A typical test data generation system was consisted by three mains aspects, including program analysis module, path selection modular as well as test data generation modular, as showed in Fig. 1.

In software testing process, program control flowchart can be derived through program static analysis and various testing path sets can be obtained depended on different coverage strategy. Only a subset of the test path set can be covered by the test data generated from stochastic test data generators. So for the residual test paths, test data generation module should be used to generate test data specially that can cover these test paths.

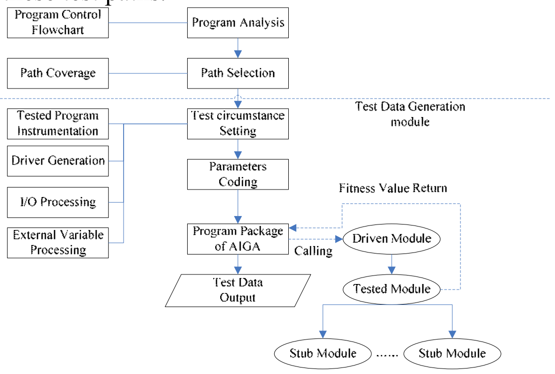

Figure 1. The structure of a test data generation system.

2) Problem Description: In software testing, the problem of test data generation can be described as follows: for the given program and a path of the program, the input space of the given program is , so the objective is to identifying the subspace , that path can be processed with as the input of program Problems of control flow test of software testing such as statement coverage, branch coverage, condition coverage, condition/decision coverage and path coverage, problems of data flow test such as definition coverage and citation coverage, problems of integrated test such as call pair coverage and data flow test and so on are all belong to this kind of problem.[6]

3) The Fitness Function of AIGA: How to definite the fitness function, which will directly affect the efficiency of the algorithm, is a key factor that affecting the performance of AIGA. A good fitness function should fully consider the background of the problem, and give a search direction to optimum solution space. For an actual problem of data generation with given test path, the goal of the algorithm is generating test data that can cover the designated path.

In this article, the fitness function SIMLARITY [9] presented by Lin and Yeh was adopted, which can measure the distance of two paths well, based on the extension of hamming distance. SIMILARITY is defined as follows:

$$SIMILARITY_{i-j} = M_{i-j}^{1} \times W_1 + M_{i-j}^{2} \times W_2 + ... + M_{i-j}^{n} \times W_n \quad (1)$$

Where:

$$M_{i-j}^{n} = 1 - \frac{S_i^n \oplus S_j^n}{S_i^n \cup S_j^n}$$

$$W_n = W_{n-1} \times | S_1^{n-1} |$$

4) Main Process of AIGA:

Step 1 Antigen inputting and parameters setting

Suitable form of solution of the software testing can be derived after the analysis of the problem and constrains on its solution, where antigen equals to the problem that to be solved through genetic algorithm, and objective function and restrictions was regarded as the input of antigen. Then the population size, crossover probability and mutation probability should be determined.

Step 2 Producing original antibodies

Identify antigen and generating the initial antibody population stochastically.

Step3 Evaluating the fitness of the antibody

In this model, the fitness of antibody and antigen can be computed by using the definition of fitness function of genetic algorithm.

Step 4 Deciding the termination condition

Whether to terminate the program was decided by reaching certain fitness or reaching designated iteration number

Step 5 Computing concentration and Expected-Breeding Ratio

The concentration of antibody was defined as the follow formula :

$$c_i = \text{the number of antibody, affinity of which is more than } \theta / N$$

(where θ is affinity constant, and generally $0.9 \le \theta \le 1$)

$$(A_b)_{ij} = 1 - \frac{\sum_{n=1}^{M} | Ab_{in} - Ab_{jn} |}{(s-1) * M}$$

Where M is the coding length of antibodies， s=10 on condition of decimal coding.

In this model, expected-breeding ratio of antibody i was adjusted as follows, which was named regulated expected-breeding ratio, after considering the testing conditions,

$$E_i = (Ag)_t * (1 + T(t) \exp(-K_{c_i})) \quad (2)$$

Where：$(Ag)_i$ is individual fitness of antibody

$T(t) = k * T(t-1)$,$T(t)$ is the annealing temperature at time t , at the beginning, the initial value of annealing temperature T0 is largest, so the concentration of antibody had obvious impaction on expected-breeding ratio that various antibodies were selected to be cloned, as the algorithm processing, annealing temperature T was becoming lower continually, and the impaction of antibody to expected-breeding ratio was descending until to no impaction. So at this case, the selection of antibody depended on concentration of antibody and annealing temperature initially, fully determined by the affinity of antigen that the antibody with higher affinity with antigen was selected to be cloned and the concentration obtained its maximum value.

Step6 Selecting

The individuals were selected to next generation using Roulette Gambling Method according to regulated expected-breeding ratio, computed in step 5.

Step7 Mutation and Crossover

Execute the operation of crossover and mutation according to crossover probability P_c and mutation probability P_m, based on the former selection of selection.

Step8 Go to step 3 after renewing the population.

III. A Case Study[10]

A. The case of test data development

In the case study, the Program of Triangle Classifier, marked as TRITYP, which was usually used as a benchmark for software testing, containing clear and comprehensive logic, was adopted as the program for test to test the efficiency of AIGA. For TRITYP, three integers denoting the length of three edges of the triangle are inputted, and the output of the program is the type of inputted triangle, that is equilateral triangle, or isosceles triangle, or scalene triangle or non-triangle. The process are as follows:

(1)Program analysis

The program flow chart and control flow chart could be obtained after analyzing the program code of TRITYP, showed as Fig. 2 and Fig. 3.

(2)Path selection

There are four executive paths in TRITYP according to the analysis of the program, which were specified in Table I.

TABLE I. THE FOUR PATHS OF TRITYP

Type	Non-triangle	Scalene Triangle	Isosceles Triangle	Equilateral Triangle
Path	Path1	Path2	Path3	Path4

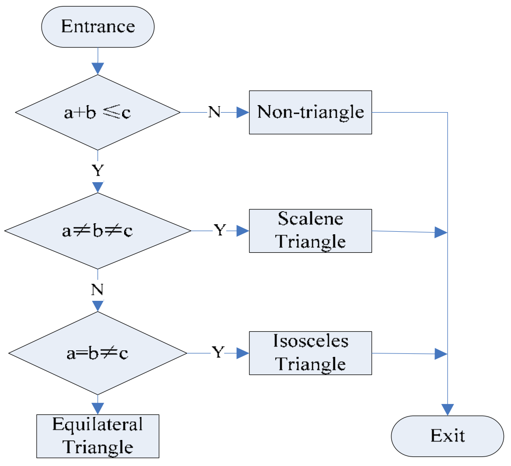

Figure 2. The program flow chart of TRITYP.

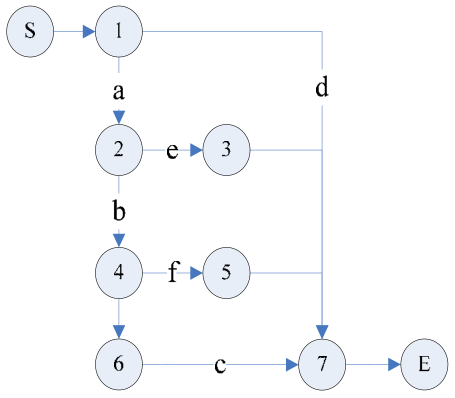

Figure 3. The control flow chart of TRITYP.

(3) Testing conditions determination

The determination of testing conditions was mainly about program instrumentation.

(4) Parameters setting and coding scheme determination

For this program the parameters were the length of the three edges of a triangle, selected from the integer set form 1 to 100. The real number coding scheme was adapted to form a chromosome (or an antibody).

(5) Initial population generation

The initial population size was 100, and then 100 chromosomes were generated randomly.

(6) Fitness fuction conputation

The fitness function SIMILARITY was adopted to evaluate the performance of each individual antibody.

(7) Termination condition Decision

If antibodies satisfy the restriction of certain fitness or iteration number has reached the designated number, then the algorithm will be terminated. In this case, the designated iteration is 20 in order to get plenty of test data.

(8) Concentration and Expected-Breeding Ratio computation

Compute the Expected-Breeding Ratio of chromosome according to the fitness of antibody and antigen. In this case, θ get the value of 1 when determine the concentration c_i of antibody i. the Initial value of temperature is 15 (or 20), and the temperature decreased by 1 degree for one generation.

(9) Selection

The antibodies were selected using roulette selection operator according to its expected-breeding ratio.

(10) Mutation and Crossover

Crossover: chromosome paired stochastically and the crossover probability P_c=0.9.

Mutation: the chromosome mutated with the mutation probability P_m =0.1.

(11) Generating the new generation of population and go to step (6)

B. *Results and Analysis*

The result of the program running based on the test data generated by AIGA automatically were showed by the real line in Fig. 4, where the transverse axis was the number of generation and the vertical axis denoted the individual number.

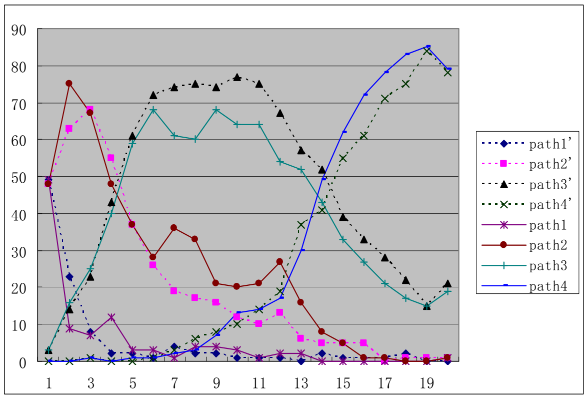

Figure 4. Test data generates by AIGA and SGA(T0＝15).

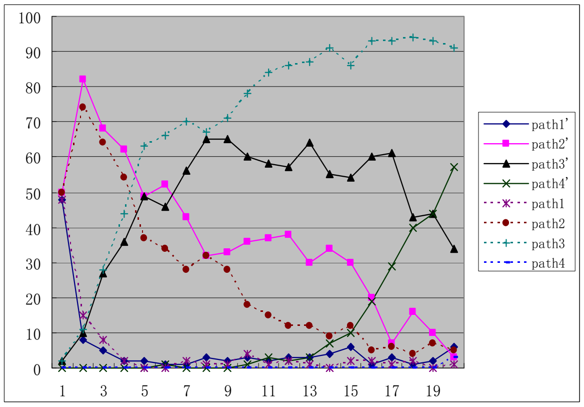

Figure 5. Test data generates by AIGA and SGA(T0＝20).

From Fig. 4, it could be got that the affinity of the objective paths, that were path4, path1, path2 and path3, increased gradually in turn. At the beginning of the program running, Path1 and Path2 were covered by the initially generated test data, that id the types of triangle contributed by the data generated were mainly non-triangle and scalene triangle. Through the operation of selection, crossover, mutation, new population was generated continually, and evolved to the direction of high affinity, that was the number of data covering Path3 were increasing while covering Path3 is decreasing. And objective path Path4 was covered by the data generated in the third generation. When the data of 20th generation were generated, the percentage of data covered Path4 reached to 79%. From the result, the efficiency was proved.

As a control, standard genetic algorithm (SGA) was adopted to generate test data with the same condition, using the initial population generated by AIGA, which was saved as an external document. The result was showed by the dotted line of Fig 3-3 and Fig. 5.

In the above figure, the fitness value of Path3 was high and its concentration increased rapidly, reaching to 70% in 7th generation. Due to the adjustment of antibody concentration, the increase of individual covering Path3 was limited, while the promotion of residual individuals was promoted. The difference of numbers of individuals generated by SGA was larger while that of AIGA much smaller. Because of the adjustment of antibody concentration, population diversity was reserved as far as possible to avoid the local convergence and promote its efficiency.

While T0 = 20, the efficiency of the adjustment of antibody concentration was more obvious as showed in Fig 3-4.

By analyzing the two groups of numbers, the conclusion that the mechanism of adjustment of antibody concentration of AIGA limited the reproduction rate to avoid the situation that individuals with high affinity constitute the main part of the population in short time. The disadvantages of SGA such as premature and local convergence were overcoming and the efficiency and performance of AIGA became more reliable and practical through reducing the selection of similar individuals to promote the performance of crossover and mutation aiming at covering more paths.

IV. CONCLUSIONS

In this paper, for the shortcomings of standard Genetic Algorithm on search efficiency, individual diversity and premature the Annealing Immune Genetic Algorithm (AIGA) was presented as the core algorithm of test data generation by introducing the mechanism of reproduction rate adjustment of individual concentration of immune algorithm and annealing principium into genetic algorithm after analyzing simple genetic algorithm, simulated annealing genetic algorithm as well as immune algorithm respectively. The efficiency and practicability of AIGA were proved by the testing of Program of Triangle Classifier, and the result was compared to that of SGA. From the result of the case, it was proved that AIGA is more efficient on maintaining population diversity than SGA, and could avoid the problem of premature and local convergence to some extent. All of the above proves that it is practical to using AIGA for software testing, which is also basement of the further research.

REFERENCES

[1] Sun Chang-ai, Jin Maozhong, “Program Instrumentation Approach to Implementing Software Dynamic Testing” (in Chinese) [J]. Min-Micro System, 2001, (12)22 1475-1479

[2] Lv Gang. Study on Immune Algorithm and Its Application (in Chinese) [D]. China University of Mining & Technology Peking.2003

[3] D.Bird, C.Munoz. “Automatic Generation of Random Self-checking Test Cases”, [J]. IBM System, 1983, 22(3):229~245.

[4] LIU Huimei, XU Huayu. “Research on Code Analysis and Instrumentation in Software Test” (in Chinese) [J]. Computer Engineering. 2007(33) 01 86-91.

[5] Bogdan Kore1. “Automated Software Test Data Generation” [J].IEEE Transactions on software engineering, 1990, 16(8):870~879.

[6] SHAN Jin-Hui, WANG Ji, QI Zhi-Chang. “Survey on Path-Wise Automatic Generation of Test Data”, Acta Electronica Sinica, 2004, 32(1):109~113.

[7] Shan Jinhui, Wang Ji et al. Path-wise Automatic Generator of Test Data and Its Graphic User Interface Design With Tcl/Tk , 2002,l:74-77.

[8] Jie Wei. Gao Zhongyi. “Research of Software Structural Test Data Generation Based on Genetic Algorithms”, Journal of Beijing University of Aeronautics and Astronautics, 1997, 23(l):36-40.

[9] D.Berndt, J.Fisher, L.Johnson et al. “Breeding Software Test Cases with Genetic Algorithms”, [J]. IEEE, 2002.

[10] Xu Xiumei. Test Data Generation Using Annealing Immune Genetic Algorithm [D]. Beijing Jiaotong University. 2007.12