http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (23): 7669-7672, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.7669.7672
© 2013 Asian Network for Scientific Information

Study on Test Data Generation Approach

[1]Wang Lu, [1]Ma Guo-Fu and [2]Cao Li-Pei
[1]Department of Computer Engineering, Anyang Institute of Technology, Henan, 455000, Anyang, China
[2]Department of Traffic Information Engineering, Henan Vocational and Technical College of Communications, Henan, 450000, Zhengzhou, China

Abstract: Automatic generation of test data can effectively reduce the cost of software development. What's more good adaptation function used in the genetic algorithm can generate a better test data. In this paper, according to the program chain in source code and the relationship between input and output data, fitness functions are respectively designed. The two are combined to form the final fitness function. To use the final fitness function to test data, the experimental results show that this method can produce test data set with high program coverage level and improve the ability to detect errors.

Key words: Genetic algorithms, data test, adaptation function, program chain

INTRODUCTION

Software testing is the key of guaranteeing software quality. It is a process which is indispensable in software development (Shen, 2001). It also is an important part of software engineering. The technique of automatic generation of test data is one of the most important technique.We can generate test examples using this technique for the program. Genetic Algorithm is a well-used method to generate the test data and its key kernel lies in the design of fitness function. The concrete way is designing fitness function based on the program structure and program criterion respectively. The generation of the fitness function based on the program structure lies in the inner structure of the program but it cannot test completely. It is difficult to design the fitness function based on the program criterion for the inexplicable understand of the inner structure and the unqualified produced test data (Pargas *et al*., 1999). In this paper, we propose a new way to produce fitness function based on the program structure and program criterion (OpenMP, 2002) together. In the result, we can use the evolve technology to produce test data set with high program coverage level and enhance the ability of detecting error.

GENETIC ALOGORITHM

Genetic algorithm produces the first generation population, maps the individual bits of the generation into actual parameter according to the way that parameter input transfers the actual parameter to the tested program, drives the tested program. The tested program calculates a branch function (evaluate function), then returns it to the Genetic Algorithm pack. Genetic Algorithm pack criticizes the merit and demerit of the bits cluster of the individual through the branch function, then generates a better genetic population by changing the structure of the individual bits by genetic operator including select, intercross and variance. Until we find the goal parameter that covers the selected path (Pargas *et al*., 1999; Zhang and Zhuang,2006).

Now we have two ways to design the fitness function. The first one is based on the program structure. Its advantage is knowing the program structure deeply. Its primal test example set can produce test data set with high coverage. But it does not work well in detecting error without considering program criterion. The other is based on program criterion. It inputs test data and performs program, then it decides the value of fitness function by program results. Its advantage is commonality. But as it ignores program structure, it is very difficult to get a fitness function to meet our requirement.

In order to dismiss the demerits we propose a method based on considering the program structure and program criterion together to design fitness function. The concrete way is designing fitness function based on the program structure and program criterion, respectively and then combining the twos as final fitness function which can enhance the ability of detecting error.

IMPLEMENTATION OF THE NEW WAY

- Design of fitness function based on program structure

Corresponding Author: Wang Lu, Department of Computer Engineering, Anyang Institute of Technology, Henan, 455000, Anyang, China

Recently the primary ways to produce test data according to source program are random method, dynamic test data method based on symbol perform and program perform, restriction set algorithm and so on. We choose the way in (Ferguaon and Korel, 1995) to produce test data according to program link. The way produces program link through analyzing the data dependence, then controls generation of test data according to the program links.

Define 1: PathSet is a set in which the paths passed are saved when test data are performed. PathSet is empty initially and the paths passed are put into the PathSet when test data are performed. When PathSet is full it contains the entire paths.

Define 2: BranchSet is a set in which the branch passed when test data perform saved. BranchSet is empty initially and puts branch passed when test data perform into the BranchSet and when BranchSet is full it contains the entire branch.

Firstly produce event sequence ES using the way in (Ferguaon and Korel, 1995), deal the Ei of the ES as follows: fetch event xi of the ES, then make xi go through event sequence Ei. If xi cannot perform when meet the node ni, it needs return to the node ni-1 and re-choose a new branch to continue. We use αi to record the number of branch xi pass from begin to ni-1, as the number of useful branch xi passed. We use βi to record the total branch Ei passed from begin to end. And use $\alpha i/\beta i$ to express the useful ratio between test data xi and event sequence Ei.

For program, tested data only go through one path. After the tested data perform, we put the path into PathSet and put the entire path of the program into BranchSet.

In order to get test data set with high program coverage level, we design the fitness function based on program structure as follows:

$$G(x) = \frac{1}{n}\sum_{i=0}^{n}\frac{\alpha_i}{\beta_i}$$

In which, n expresses magnitude of the Event Sequence (ES), x expresses one test data, $\alpha i/\beta i$ expresses the useful branch ratio between test data x and event sequence Ei. Obviously, the bigger the G(x) is, the higher the useful branch ratio.

- Design of Fitness Function based on program criterion

Program criterion gives thecorresponding relation between input data and output data, so we can analyze the output according to input and which can help us see whether error exists in program or not. We compare test data output value Output(x) with expected value True(x). If there exists something different, it can identify that there are something out of criterion. Then calculate the value H(x) of fitness function based on program criterion according to the branch the test data passed. If the two equals, it can identify that this test data is no use to the fitness function based on program criterion and H(x) is 0.

Define 3: Function R(x) is a stochastic function, its range is from 0 to x.

We quote the equivalence class to optimize the test data. Equivalence class is test data in the same class, where if one test data in the equivalence class has errors, others in the equivalence class can also find the same errors (Tahat *et al*., 2001). If test data A and B pass the same path, they are in the class with high possibility. The more both A and B pass the same path, the higher possibility they are in the same class. The way to optimize is that choosing only one test data in per equivalence class. If test data A and B are in the same category with a high possibility, the value of fitness function based on criterion will be small.

Case 1: If the paths that test data A passed have existed in PathSet, it means that there were test data passed the same path with A. Now the value of fitness function of A is low. We define the value of fitness function of A as H(x) = R(1/4).

Case 2: If the paths that test data A passed have not existed in PathSet, it means no test data has passed the same paths with A. If the branch that A passed is in BranchSet, it means that test data A has not passed a new path. We define the value of fitness function of A as H(x) = R(1/2).

Case 3: If at least one of the branch that A passed is not in BranchSet, it means that A has passed a new branch that the other test data doesn't pass. So A has a high possibility to find error and the value of fitness function of A is high. The value of fitness function of A we define as H(x) = R(1).

- Design of final fitness function

The fitness function is to be designed as $F(x) = \alpha \times G(x) + \beta H(x)$.

It manifests the trait of the method proposed in this paper combining the program structure and program criterion. G(x) and H(x) is the branch of fitness function based on program structure and program criterion respectively, α and β are modulus and their value range is from 0 to 1. Then we can know that if $\alpha = 0$, the method is actually based on program criterion, if $\beta = 0$, the method is actually based on program structure. The values of α and β are inputted by tester for adjusting extent of the two methods. We can see from the experiment that adjusting of the two is very important to the final results.

DESIGN OF EXPERIMENT AND ANALYSIS OF RESULTS

We choose the Dijkstra algorithm program as our test program. The program has 155 rows, 83 nodes and 51 branchs. We add some errors to the right program to validate the error detecting capability of the method and then we will know the proportion of error detecting. Here we introduce "variance" in which the called "variance" is introducing many transformative edition according to one right edition, then add errors into the transformative edition. When one program is performed (Offutt, 1995), we compare the program results of transformative edition with the results of the right edition to see whether it has detected errors or not. At last we analyze the average value of each index such as coverage ratio, error detecting proportion etc to testify feasibility and superiority of the method proposed in this study.

Firstly genetic algorithm produces primary test data stochastic as generation population, where one of the transformative editions of the program to be tested is drived by one driver, then inputs the primary test data produced stochastic into the transformative edition to perform, after n generations of evolution (n = 500 in this study) or the fitness function is convergent (the gap between the values of fitness functions of the next two generation is less than δ and δ is defined as a minimum). The test data produced is what we require then we calculate the coverage ratio, proportion of detecting of this group test data facing the transformative edition.

From the experiment we get the result as follows: α and β are modulus of the fitness function. Experimental environment: OS is Windows2000, Intel Core 2 Duo, RAM is 1G, the number of transformative edition performs is 40.

From Table 1 we can see that the branch coverage ratio can achieve a high value and path coverage ratio is 85.4. If program is complicated, the path coverage ratio will be low and foremost the proportion of detecting errors is 97.1%. The result manifests that this method impacts well on detecting error. We also find that the values of α and β affect the generations of the convergence of fitness function mostly.

In the experiment, when $\alpha = 0.6$ and $\beta = 0.4$ the proportion of detecting errors achieves the biggest and the generations of evolution also achieve the biggest. So how to adjust the values of α and β to get a more excellent result is the question we need solve next. As can be seen in Table 2, we compare the result produced when $\alpha = 0.6$ and $\beta = 0.4$ with test data produced automatically by other methods.

From Table 2 we can see that randomized trial is simple but the effect is worst. The average of program coverage ratio got from method in paper (Ferguaon and Korel, 1995) and paper (Korel and Al-Yami, 1996) is higher than our method but for proportion of detecting our method is better. Paper (Korel and Al-Yami, 1996) can get a high proportion on detecting error, because it adopts a way of inserting value in program (insert assert in program). If the result of the program is out of criterion, it manifests that program is wrong at inserted place. Ferguaon and Korel (1995) achieves a good program coverage ratio but its ability of detecting error is not sufficient, especially when it deals with the boundary value because it only analyses the program structure but program criterion. The purpose of testing is to detect

Table 1: Result analysis

α	β	Branch coverage ratio (%)	Path coverage ratio (%)	Proportion of detecting (%)	Generations of evolution
0.2	0.8	97.9	74.4	94.1	360
0.3	0.7	97.5	76.3	95.2	373
0.4	0.6	98.4	79.6	95.8	376
0.5	0.5	97.1	79.4	95.5	386
0.6	0.4	99.0	84.3	97.2	411
0.7	0.3	97.8	85.1	95.2	363
0.8	0.2	97.6	85.4	92.6	351

Table 2: Method compare

Capability	Randomized trial (%)	Method in (Ferguaon and Korel, 1995) (%)	Method in (B. Korel and Al-Yami, 1996) (%)	Our method (%)
Path coverage ratio	66.3	88.4	86.3	84.3
Proportion of detecting	62.5	83.1	91.5	97.2

errors but not to achieve high program coverage, though high program coverage can help us detect error. For this our method is better.

SUMMARIES

Genetic algorithm has a broad foreground in field of automatic generation of test data. Aiming at the deficiency of the evolve method of test data, we proposed a new method to design fitness function based on program structure and program criterion. The concrete way is designing fitness function based on the program structure and program criterion respectively and then combining the twos as final fitness function. The mostly trait of this method is enhancing the ability of detecting error. Our experiment testifies the validity of our method. We can adjust the proportion of program structure and program criterion in fitness function by adjusting the values of α and β. But we don't know what values of α and β can introduce to the best result and this is our futher work.

REFERENCES

Korel, B. and A.M. Al-Yami, 1996. Assertion-oriented automated test data generation. Proceedings of the 18th International Conference on Software Engineering, March 25-29, 1996, Berlin, Germany, pp: 71-80.

Ferguaon, R. and B. Korel, 1995. Software test data generation using the chaining approach. Proceedings of the International Test Conference, October 21-25, 1995, Washington, DC., USA., pp: 703-709.

Tahat, L.H., B. Vaysburg, B. Korel and A.J. Bader, 2001. Requirement-Based automated black-box test generation. Proceedings of the 25th Annual International on Computer Software and Applications Conference, October 8-12, 2001, Chicago, IL., USA., pp: 489-495.

OpenMP, 2002. OpenMP C and C++ application program interface. Version 2.0, March, 2002, http://www.openmp.org/mp-documents/cspec20.pdf.

Offutt, A.J.A., 1995. A practical system for mutation testing: Help for the common programmer. Proceedings of the International Test Conference, October 2-6, 1994, Washington, DC., USA., pp: 824-830.

Pargas, R.P., M.J. Harrold and R.R. Peck, 1999. Test data generating using genetic algorithms. Software Test. Verific. Reliab., 9: 263-282.

Shen, C.S., 2001. Design of software test example. Inform. Micro Comput., 2: 48-49.

Zhang, K.D. and Y.B. Zhuang, 2006. Tutorial of Engineering of Software and Software Test. Publishing House of Electronics Industry, China.