2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Eccentric Test Data Generation for Path Testing Using Genetic Algorithm

Punam Mishra[1], Bhabani Shankar Prasad Mishra[2]
[1] Lecturer, Dept. of Computer Science, KISD, Bhubaneswar
email- Punam_mishra16@rediffmail.com
[2] Lecturer, CSE Department. Campus-3, K.I.I.T University, Bhubaneswar
email-mishra.bsp@gmail.com

Abstract. Effective and efficient test data generation is one of the major challenging and time-consuming tasks within the software testing process. Researchers have proposed different methods to generate test data automatically; however, those methods suffer from different drawbacks. In this paper we present a genetic algorithm-based approach that tries to generate a test data that is expected to cover a given set of target paths. Our proposed fitness function is intended to achieve path coverage that incorporates path traversal techniques, neighborhood influence, weighting, and normalization. This integration improves the GA performance in terms of search space exploitation and exploration, and allows faster convergence.

Keywords: Test data, Software testing, Genetic algorithm, Path traversal

1. Introduction

Testing is a very tiring, tedious and time consuming process. It is treated as a parameter for software quality assurance and represents the final review of specification, design, and coding. It also ensures that the software runs correctly as per customer requirement. Software testing accounts for 50% of the total cost of software development [1]. This cost could be reduced if the process of testing is automated For software testing, test data have to be generated and prove them as better than other test data. Therefore, a systematic testing system has to differentiate 1-2 good (suitable) test data from bad test (unsuitable) data, and so it should be able to detect good test data if they are generated. Nowadays testing tools can automatically generate test data which will satisfy certain criteria, such as branch testing, path testing, etc.

In path testing every possible path in the program is tested. A path is nothing but set of conditions through which input travels from the starting point to the exit point. However, path testing is generally considered as a infeasible task because a program with loop statements can have an infinite number of paths. The path testing problem is considered as NP-complete problem, various test data generation methods for path testing have been proposed in the literature [4].

In this paper a genetic algorithm based approach has been proposed. The method is a metric which calculates the difference between the selected path to be traversed and the actual path taken by the input. A genetic algorithm tries to find the test data which traverses a given selected path is as follows:-

2. Proposed Algorithm

Euclidean distance [11] is taken to quantify the distance between two paths of the control flow graph. It is actually a metric which calculates the differences between two given path ie the target path and the exercised path. If there are branches for a particular path, then the testing may fail because two different paths may contain some common edge in different sequence. Therefore extended mode of the Euclidean distance is used. it is derived from Pythagoras theorem. Euclidean distance in two dimensional forms is given by:

Let P and Q be two points in Euclidean space

$P=(p_1,p_2,\ldots\ldots\ldots p_n)$

and

$Q=(q_1,q_2,\ldots\ldots q_n)$,then distance is calculated as:

$$\sqrt{(p1-q1)2+(p2-q2)2+\ldots\ldots+(pn-qn)2}=\sqrt{\sum_{i=1}^{n}(pi-qi)n}$$

Normalized Euclidean distance is used so that values will always lie between [0,1].

Normalized Euclidean distance = Euclidean distance/range [12]

Now we can define a fitness function CLOSENESS=1-(normalized Euclidean distance)

1.1 Cascaded branch testing:

Let G= control flow graph

$P=\{path_i \mid path_i$ is a complete path within $G\}=\{path_1, path_2, path_3\ldots\ldots, path_z\}$

N=number of complete path of p.

Let

$S_1^{\ 1}=\{g{:}g$ branch of path1$\}$

$S_1^{\ 2}=\{h{:}h$ is a cascaded branch of path1$\}$

$S_1^{\ n}=\{k{:}k$ is a n-tuple cascaded branch of path1$\}$

Continuing like this……,

.Sqt={r:r is a t-tuple cascaded branch of path q}

First order distance:

Di-j1=√(x1-y1)2+(x2-y2)2

Ni-j1=Di-j1 ⁄range

D i-j 2=N i-j 2=D i-j 2 /range

Continuing like this……..,

.D i-j m=N i-j m=D i-j m /range

M i-j m=mth order closeness between i and j

M i-j m=0 or N i-j m=1 if path i and path j have no common branches

When N i-j 1=1,N i-j 2=1…………N i-j n=1

or

M i-j 1=0,M i-j 2=0……………..M i-j n=1

It results in worst test case generation

When N i-j 1=0, N i-j 2=0…………N i-j n=0

Or

M i-j 1=1, M i-j 2=1……………..M i-j n=1

It results in best test case generation

CLOSENESS can be computed as:

M i-j 1*w1+ M i-j 2*w2+ M i-j 3*w3+……… M i-j n*wn

Where w=weighing factor which are chosen with experience

If CLOSENESS i-k >CLOSENESS i-j, then path k is much closer to target than path j.

3. Basic steps of path testing

1. Construction of control flow graph
2. Target path selection
3. Test data generation from the genetic algorithm and execution [6].

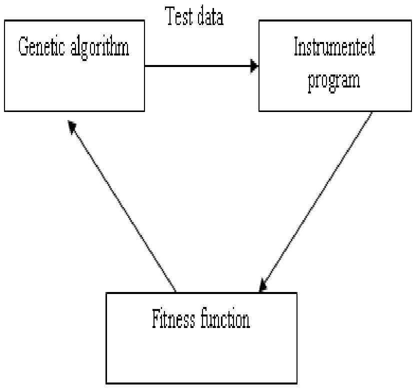

Fig. 1: Basic steps of test data generation.

For the testing process, triangle classifier problem is taken as a problem. the c code for triangle classifier problem is as follows:

```
# include<string.h>
Char path[256];
Int triangleA(unsigned int a,unsigned int b,unsigned int c)
{
        int triangle=0;
        if((a+b>c)&&(b+c>a)&&(c+a>b))
        {
                strcat(path,"u");
                If((a!=b)&&(b!=c)&&(c!=a))
                {
                        strcat(path"y");
                        Triangle=1;
                }
                else
                {
                        strcat(path,"v");
                        if(((a==b)&&b!=c))||((b==c)&&c!=a))!!((c==a)&&a!=b)))
                        {
                                Strcat(path,"z");
                                Triangle=2;
                        }
                        else
                        {
                                strcat(path,"x");
                                Triangle=3;}
                        }
                }
        }
        else
                strcat(path,"w");

return(triangle);
}
```

Control flow graph construction:

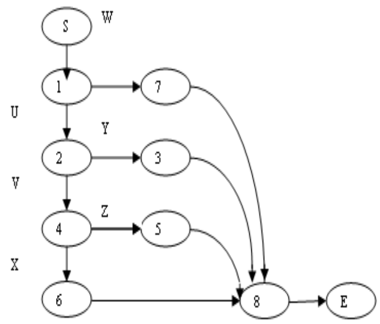

Fig. 2: Control-flow graph.

4. Conclusion

From the above discussion about genetic algorithm, we can conclude that genetic algorithm is used to generate test data automatically. The greatest merit of using the genetic algorithm is its simplicity. The proposed fitness function CLOSENESS is very simple and can enhance the speed.

5. References

[1]. B.Beizer.Software Testing Techniques,Van Nostrand Reinhold,2nd edition,1990.

[2]. B.Korel.Automated Software Test data Gneration.IEEE Transaction on Software Engineering 16(8);870-879,August 1990.

[3]. Jon Edvardsson. A survey on automatic test data generation .In proceedings of the second conference on computer science and engineering in Linkopinng, pages 21-28,ECSEL,October 1999.

[4]. Bruno T.Abreu,Eliane Martins,Fabiano L.deSousa.Automatic test data generation for path testing using a new stochastic algorithm.Supported by Brazillian CNPq council through Reasearch Grant 134107/20047

[5]. Roy P.pargas,Mary jean Harrold,Robert R.peck.test data generation using genetic algorithm.journal of software testing,verification and reiiability.1999,wiley copy right.

[6]. lin,j and yeh,p.(2001).using genetic algorithm for test case generation in path testing. information sciences,vol-131,2001 pp 47-64.

[7]. philmcminn.Search-based software test data generation.A survey published in software testing,verification and reliability 14(2),pp 105-156 june 2004,copy right(c) wiley 2004.

[8]. Sthamer ,H(1996).the automatic generation of software test data using genetic algorithm,phd thesis,university of glamorgan pontyprid,wales,Great Britain.

[9]. D.goldberg.genetic algorithm in search,optimization and machine learning.addision-wesley, reading,Massachusetts,1989.

[10]. Yong chen,Yong Zhong.automatic path-oriented test data generation using a multipopulation genetic algorithm,Fourth International conference on Natural computation.IEEE computer society 11-566-569.

[11]. F.fessant,P.Aknin,L.oukhellou,S.Midenet.Comparision of supervised self organizing maps using Euclidean or Mahanalabis distance in classification context ,6th international work conference in artificial and natural network(IWANN 2001),Granada,june 13-15,2001.

[12]. Killian q.weinberger,Lawerence Kaul. Fast solvers and efficient implementation for distance metric learning.preliminary work under review by the international conference on machine learning.