ISSN No. 0976-5697
Volume 4, No. 10, Sep-Oct 2013
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Introducing a New Approach to Produce Software Testing Data by use Selecting Procedures in Genetic Algorithm

Farideh Keshavarz
Department of computer, Sirjan branch,
Islamic Azad University, Science and research
Sirjan, Iran
faride.ke@gmail.com

Reza Nourmandi-Pour
Department of computer, Sirjan branch,
Islamic Azad University,
Sirjan,
noormandi_r@iausirjan.ac.ir

Arash azizi mazrae
Department of computer, Sirjan branch,
Islamic Azad University, Sirjan,
arash_azizim@yahoo.com

Abstract: Making sure about the truth of program performance is one of the challenging issues discussing by software engineers in this order the program should test with different and complete data so a set of data is needed to test the program entirely. This is so complicated for big and complex programs so in software engineering there is a branch so-called software engineering test in this paper software is performed by genetic algorithms. As there algorithm work randomly band produce test data which catching them may will be difficult in this paper we mentioned the history (profile) of software test technics and genetic algorithm and finally mention suggested approach and results of simulating using MATLAB software.

Key words: Covering path, genetic algorithm, software testing data, software testing

I. INTRODUCTION

Software validating and viewing (v&v) is set of inspecting processes and analyze the truth of software and manufacturing the software and match it to clients' demands[1, 2]. These two (v&v) are available through the software life and start with needs continue till designing code and testing process Boehm knows viewing as a respond to this question that : whether we are making the truth(correct) product and validating answer to this question : whether we are making the product correctly? [3, 4, 5] These two definition shows that viewing software includes inspecting the match of downloads to its description and validating is a more total process which it should make sure the a need of clients both quality is accepted in other words [6]. software validating is answer to this question that when the software goes to real world if it responds correctly and does its own duly well software because of environment emergency such as faults or code errors may be call apse five total goal of viewing and validating (v&v) are:

a. Truth, software includes no fault
b. Matching means software is match able itself
c. Necessity, everything in software is necessary
d. Complete, all of its demands are catchable
e. Performance, clients' needs are done efficiency

In order to this to do viewing and validating software static viewing and dynamic viewing have been used to view and analyzing demands- designing middles and program code are using but dynamic viewing is done when program is running on a machine to inspect results matching of running a program and its expected results[7, 8].

Software test is research the quality of a product or software service and output the information to clients this research includes software to find its errors but not restricted to this tests are a questionnaire which the software is tested by it while it is expected from program that output the correct answers [9].

II. SOFTWARE TESTING HISTORY

The main purpose of designing software testing is to collect a set of tests which is more likely to discover software errors in order to this two different technics are used: white box test and black test box white box test is based on controlling structure and it is to make sure all the program commands are runnel once at least and all logical condition are tested. Basic path test is a white box technic uses graphs' to achieve set of linear tests so guarantee the cover of all cases test condition, data flew and program logic and loops with preparing a procedure to run loops with variety of complete technics in white box [10, 11].

Hetzl described white box as test in small dimensions. He means that while box test usually used for small parts of a program on the other hand black box test works in a more vested area which can called it test in macro dimension black box test used to validity of needs without core to inside the program [12, 13, 14].

Black box technics emphasize on information domain in this order by dividing the domain of input and output of a program the data for test is being ready- when dividing the input domain divided to levels of data so practice by software is more likely to do boundary data description enable the program to control data in boundary. Array testing is an efficient and systematic method for systems which its input parameters are very low. The other testing method includes vast area of software abilities such as graphic interface- clients' architecture, server, documents and guidelines, real time systems that each needs especial commands to run adroit programmers often mentioned that the testing never ends just you (software engineers) transfer to client to clients. Every time clients uses a program a test is done software engineers using designing test can reach a completed test and so most of the errors before clients test has been detected [15].

III. GENETIC ALGORITHM HISTORY

This algorithm is part of eventually calculating that recently known as a part of artificial intelligent (A.I) – the main idea of this algorithm is concealed inside the Darrin evolution theory practically genetic algorithm is one of the optimizing methods that is based on natural selection and some aspects which inspirited from genetic science. Generally genetic algorithm: includes, chromosome, population, operators function some of their application includes optimizing automatic programming, machine learning, eventually evolution, ecology and social systems [16, 17].

Also in genetic algorithm many coding for different domains used such as binary coding, tree and selecting, many methods are suggested for genetic algorithm for example matching selection, SUS method, and paragon [18].

IV. SUGGESTED OUTLINE

In a program testing with covering index a suitable if cause parsing a path is a set of mane starting from a nod to last nod in a graph and an in depended paths is a path that at least one of its main has not been met by other paths in software testing main challenge is producing automatic and disciplined data a sufficient and necessary data is a data that only and if only cause an independent parsing- in this part based on data flew graph and to cover all manes one mane is called critical if in one path it caused in dependent path so critical manes in a path means there is no need to search other mane in paths and searching is from o(2n) nearer to o(n) because every mane is for a linear and independent path.

Suggested method includes:

a. Finding critical mane for data control flew graph (the inputs)
b. Determining fitness function
c. Determining critical manes covering table
d. Determining first data (chromosomes)
e. Calculating covering manes using cover table
f. Generation product and evaluating chromosome
g. Determining chromosome and replace by optimized once
h. Recovery outputs and covering table

The steps 5-8 are repeating that its stop up to operator or time consumed. Operator ideas is asked him/her based on results of steps 5, 8 to find critical manes firstly the graph divided into levels that each has a simple branch or a complex branch- then for each level the below operations are done:

a. Choose the in rest branch and call its mane as critical mane cross it put it in critical manes sets
b. The rest of manes also crossed this path starts from beginning level nod
c. If there is uncrossed mane it should remove to level 1- critical manes are representing all graph so by inspecting a percent of graph can be evaluated in fact this evaluation determines commands covering now a table called covering table is composed for critical manes and their commands using. This table a stop index for generation production in genetic algorithm will be created which this indexes the number of parsed path which are near to McCabe number present.

Covering table as get critical edges as data or random data that so- called presenting chromosomes or firstly chromosome of genetic algorithm. Because evaluating function in genetic algorithm has basically role this function has basic role in production optimize data this function is introduced as below and evaluates the condition related to critical edges

a. Production the program control current graph and determine the branches related to critical edges and finding condition commands
b. Determining critical edges
c. determining logical pharoses up to conditional commands from last step and making logical sentences using the commands such as "if" and "case" which parallel excursion is not possible so if one part of "then" and "else" and "if" are part of a program the combining of this is for making a logical sentence and for other combination use logic commands
d. Determine evaluating function from logical sentences from step3

a) 4-1 each combing operator is replaced by minimum function or maximum function
b) 4-2 each condition operator is replaced by equal function

$$if(x) \qquad f(x) = \begin{cases} 0, & true \\ K, & false \end{cases}$$

$$if(x = y) \qquad f(x) = \begin{cases} 0, & x = y \\ abs(x - y), & x \neq y \end{cases}$$

$$if(x \neq y) \qquad f(x) = \begin{cases} 0, & x \neq y \\ K, & x = y \end{cases}$$

$$if(x > y) \qquad f(x) = \begin{cases} 0, & x < y \\ (x - y) + K, & x \geq y \end{cases}$$

$$if(x \leq y) \qquad f(x) = \begin{cases} 0, & x \leq y \\ (x - y), & x > y \end{cases}$$

Figures 1. Fitness functions for conditional relationship.

First step the data which are created randomly or by operator compose represented chromosome or their binary equal and make chromosome gens then by initial (corms) execute program and for every critical edges determine it by "y" in covering table second step starting of population production loop in this loop for each data the function combined or jumping are called then evaluating function use parsed edges evaluated the data so next generation is created the more quality the stronger chromosomes (corms) and better chances for next loop may be optimized corms in current generation will not be created in next loops they will be saved in "op ch2" variable if the quality of a corms is lower than a minimum that corms is collapsed and it will be replaced by optimized corms from last loop and at the end of loop the number of parsed edges covering table calculated and this number by covering table showed to operator forth step if operator request to execute an algorithm in a loop the "counter" variable also repeated till not more than a value.

Procedure GenC (){

```
Input:  Program: Changes version of program to be tested;
InitData: Set of test data; CE: Crucial Edges;
Output: Final: A solution test case set;
CTable: recorded CEs with status;
#define  MaxTimes  max;  //Max acceptable time;
#define MC acceptable number of McCabe number;
Variables declaration:
CCH1: Candidate chromosomes;
CCH2: Candidate chromosomes;
TCE:Traversed Crucial l Edge;
CTable: Coverage Table;
NextP: a set of test data;
CPop: a set of test data;
OCH1: Optimal Chromosomes;
OCH2: Optimal Chromosomes;
Cnt: iteration;
Begin
Level1:
        Make CCH1& CCH2 by InitData;
        Get fitnessFUN() to Initial OCH1 and CTable;
 Initial CPop;
Level2:
        While (! fill CTable with Y || counter <
MaxTimes ||! Reach MC ||! User request) {
Use Crossover and Mutation operations;
Compute fitnessFUN ();
Compute NextP and Save OCH2;
Level3:
        for each chromosome of NextP
If (Defect (NextP))
Replace with OpCH1 one;
CPop = NextPop; OCH1 = OCH2;
Level4:
        if(cnt mod 10 == 0){
Compute number of TCE by CTable;
Show CTable and Ask to continue;
}
cnt++;
}
Final = CPop;
Return Final and CTable;
End.
}
Boolean Defect (chrom){
val = Fitness value of best OPCH1;
if fitness value is less than v/l return true;
else return false; //l is an optional value
}
```

The recommended genetic algorithm has 4 stopping index:

a. parsing all critical edges (that after searching covering table it is cleared
b. the near distant between parsed path to McCabe number
c. Time restricted (it is based on algorithm time and if it is more than the output will decreased)
d. Operator Satie faction (after each execute of algorithm or covering table the number of covered path ask operator if happy stop the algorithm)

V. CONCLUSION

White box test is based on program structure so it is not applicable for programs that their source or codes not access able and should use black box test in comparison as this method is based on critical edges the covering table will be smaller and simpler so in a result the path will change from ultimate degree to linear degree and with saving optimized chromosome in every step the algorithm is patched to go faster in this method in every step the operator aware of progress and if after many loop of algorithm it will be cleared that path covering is not parsed it means there are some paths which never will parsed that so called impossible path i.e. there is no data to parse them so the operator will be awarded and stop the algorithm in order to qualified the progress some other algorithms can replaced by genetic algorithm which it needs the dimension of new problem will match to new algorithm so its character can be used in problem solving at the end the result will compare and find another recommend for better ability an algorithm to solve problem and combine algorithm to find better results continually current graph to merge 2 arrays of searching then the chromosome table firstly and lastly will be mentioned.

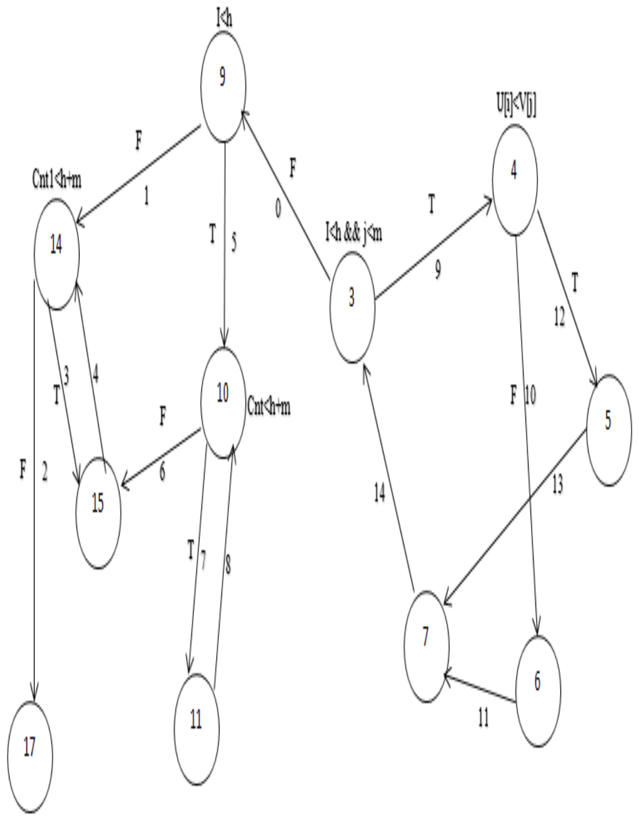

Figure 2. Program flow graphs merging two sorted arrays.

Table 1. Edges in the graph in Figure 2 instead of two chromosomes

Edges	CH1	CH2
0	1	1
1	1	1
2	1	1
3	2	2
4	2	2
5	0	0
6	0	0
7	0	0
8	0	0
9	2	2
10	0	0
11	0	0
12	2	2
13	2	2
14	1	2

Table 2. Edges in the graph in Figure 2, the final chromosomes instead.

Edges	CH1	CH2	CH3	CH4	CH5	CH6
0	1	1	1	1	1	1
1	0	1	1	0	1	1
2	0	1	1	0	1	1
3	0	2	1	0	2	1
4	0	2	1	0	2	1
5	1	0	0	1	0	0
6	1	0	0	1	0	0
7	2	0	0	1	0	0
8	2	0	0	1	0	0
9	2	2	3	2	2	3
10	0	2	2	1	2	2
11	0	2	2	1	2	2
12	2	0	1	2	0	1
13	2	0	1	2	0	1
14	2	2	3	3	2	3

VI. REFERENCE

[1] Sommerville. Software Engineering, 8th Edition, Addison Wesley, 2006.

[2] B.W. Boehm. Software Engineering, R&D Trends and defense needs. In Research Directions in Software Technology, Cambridge, MIT Press, 1979.

[3] CMMI for Systems Engineering/Software Engineering ,Version 1.02 , 7 CMU/SEI-2000-TR-028, Software Engineering Institute, 2000.

[4] W. Myers. Can Software for the Strategic Defence Initiative ever be Error-Free? IEEE Computer, 19(11), Nov. 1986.

[5] R.K. Iyer and P. VerLardi. Hardware-related Software Errors: Measurement and Analysis, IEEE Transaction on Software Engineering,11(2), Feb. 1986.

[6] J.P. Queille and J. Sifakis. Specification and verification of concurrent Systems in CAESAR, In Proc. Of 5th ISP, 1981.

[7] A.J. Offutt and L. Shaoying. Generating Test Data from SOFL Specifications, The Journal of Systems and Software, volum 49, issue 1, pp. 49-62, Elesevier Science Inc.,1999.

[8] G.J. Myers. The Art of Software Testing, Second Edition, John Wiley, 2004.

[9] M. Mitchell. An Introduction to the Genetic Algorithms, MIT Press, Cambridge, MA, 1996.

[10] D. Beasley, D. Bull and R. Martin. An Overview of Genetic Algorithms: Part 1, Fundamentals, University of Cardiff, Cardiff, 1993.

[11] C.Ghezzi, M.Jazayeri and D.Mandrioli. Fundamentals of Software Engineering, 2nd Edition, Prentice-Hall, 2003.

[12] A.H.Watson, T.J. McCabe. Structural Testing: A Testing Methodology Using the Cylomatic Complexity Metric, Computer Systems Laboratory, National Institute of Standards and Technology Gaithersburg, MD 20899-0001, 1996.

[13] T.Manterea, J.T. Alander. Evolutionary software Engineering, A Review, Journal of Applied Soft Computing, 5(3), PP. 315-331, Elsevier, 2005.

[14] J.Miller, M.Reformat, H.Zhang. Automatic test data generation using genetic algorithm and program dependence graph, Journal of Information and Software Technology, Elsevier, 48(7), pp. 586-605, 2006.

[15] M.A.Ahmed, I.Hermadi. GA-based multiple paths test data generator, Journal of Computers & Operations Research, Elsevier, 35(10), pp. 3107-3124, 2008.

[16] S.N. Sivanandam, S. N. Deepa. Introduction to Genetic Algorithms, Springer, 2007.

[17] J.Miller, M.Reformat, H.Zhang. Automatic test data generation using genetic algorithm and program dependence graph, Journal of Information and Software Technology, Elsevier, 48(7), PP. 586-605, 2006.

[18] M.A.Ahmad, I.Hermadi. GA-based multiple pathes test data generator, Journal of Compuers & Operations Research Elsevier, 35(10), PP. 3107-3124, 2008.