ISSN: 0976-3104
ISSUE: COMPUTER SCIENCE

ARTICLE

GABVIE MODEL FOR SOFTWARE TESTING

Seema Sharma*, Shaveta Bhatia
Faculty of Computer Applications, MRIIRS, Faridabad, INDIA

ABSTRACT

Software testing is an immensely important topic, which helps us to create robust and effective software. Software testing means to compare the expected outcomes and real outcomes to ensure that the software product is free from error. The software testing is essential as the software bugs could be costly and risky. The testing by considering the code is referred to as the white box testing. There are many factors that should be consider before testing the software like statement coverage, decision coverage, branch coverage, condition coverage etc. This paper proposes a novel and effective model to carry out white box testing by finding the importance of a variable of the program code using Genetic Algorithms. The model is promising and would therefore help both the researchers and practitioners alike. The fitness function in Genetic Algorithm has been calculated mathematically which outperforms the earlier traditional methods. It includes the information necessary for ascertaining the importance of a variable.

KEY WORDS
Software testing, Genetic Algorithm, Fitness Function, White box testing.

INTRODUCTION

Software testing has been conducted with an intention to find bugs and verify whether software is fit for use or not. It provides the information of quality of software to the stakeholders. Software testing is conducted for the analysis to check.

1. Whether it meets all requirements to design and development of the product
2. All types of inputs must be responded or not
3. All functions perform well or not

Software testing is very complicated task. It accounts for the major portion of the project time [1]. It can be classified as White box and Black box testing [3]. White box testing explores the code while the black box testing only sees the input and output [4]. The grey box testing lies in between them. Though many techniques are available for the efficient and effective testing of a program, as per the literature review, they do not consider the occupancy of a variable in a program [2]. This research has taken the above limitation by adding the parameters which are "variables" used in code for appropriate testing taken as the initial population in GA.

In this paper; a novel model is proposed to accomplish white box testing. The work uses the frequency of a variable used in the program code by calculating its dynamic priority based on its occurrence [2]. Furthermore, the fitness of a variable is calculated mathematically based on the rate of recurrence. Since the search space can become very large, therefore a heuristic search technique is needed to access the importance and hence to select a variable. A heuristic technique is fast calculation method for getting the optimal solution. Genetic Algorithm is a heuristic search process which is based on the survival of the fittest [19-22]. The explanation of the algorithm has been included in the paper.

Received: 26 Sept 2019
Accepted: 15 Oct 2019
Published: 20 Oct 2019

The work is organized as follows. Section 2 of this paper presents a brief literature review. Section 3 presents a discussion on Genetic Algorithms. Section 4 presents the proposed work and algorithm and the last section concludes.

LITERATURE REVIEW

There has been immense work has been done on software testing by various researchers. But still to develop the better model for testing the software. The proposed work deals with developing the model used to test the program code which is discussed in the next section. Some of the related works done by the prominent researchers and the issues in this field are discussed below. A new strategy has been proposed on the basis of existing work.

The combined features of Cuckoo search and genetic algorithm has been proposed by the authors Khan et al. in 2018 [14] for optimization of test cases. A cuckoo search algorithm first ameliorates arbitrarily generated test cases and after that, genetic algorithm is used to create new test cases. There is a need of more improvement in their algorithm in order to get efficient and optimized result. Also the cuckoo search algorithm is applied only for the amelioration of test cases and genetic algorithm is applied for the new generation of test cases, which increase the complexity of the algorithm. Bashir and Nadeem in 2017[10] proposed a technique for generating the test case for mutation testing. Since mutation testing is time consuming and expensive task therefore the author has taken genetic algorithm approach for reducing the cost. A new fitness function was introduced; eMujava and the results were compared with standard fitness functions. The authors have taken four parameters for testing fitness function for state based and object oriented program, two ways cross over and adaptable mutation. However, other versions of cross

***Corresponding Author**
Email:
seema.fca@mriu.edu.in
Tel.: +91-9873553830

IIOAB JOURNAL
ISSN: 0976-3104

over and mutation have not been taken.

A new approach have been implemented for the automatic test case generation in order to improve the efficiency of software by Khan and Amjad (2015) [13]. In this paper the author suggested that random method is inadequate for the selection process of test data. For this purpose the author used mutation analysis and genetic algorithm approach to produce test data for program code. The inter procedural control flow graph (CFG) is used for the program. This weight of CFG is distributed among all edges according to the pareto principle. However the algorithm introduced is suitable for unit testing but if we want to test the overall code then mutation testing is not effective. The Markov Chain approach with genetic algorithm has been applied by the authors Boopathi et al. 2014 [11] for testing the software. In his approach, the code is firstly converted into control flow graph and then optimized to implement DD-graph. The fitness function is based on coverage of path in DD-graph. The weights of edges are assigned with the help of Markov transition probability matrix and the fitness function of genetic algorithm is calculated as the sum of all the weights of edges in DD graph. The operators of genetic algorithm are applied for generating the test cases to cover maximum path. Only one point crossover and uniform mutation genetic operator is used. The results are not compared with simple genetic algorithm or genetic algorithm with varying population or with random techniques.

Saini and Tyagi (2014) [5] have used two different optimization techniques, Genetic Algorithm (GA) and Clonal Selection Algorithm (CSA)for test case generation. This paper presents to initiate the test data by using these techniques to test the basis path testing. Korel's Distance Function (Korel 1990) is used for accomplishment of fitness function. Since basis path testing means every statement and every branch should be test. But fitness function used examined only branch predicates. Various experiments have been evaluated in Matlab like square root, quadratic equation, trigonometry function, Linear equation etc and compare the results with the random-testing technique. Also using these techniques we cannot conclude, which algorithm is best GA or CSA. Minj and Belchanden (2013) [3] had presented the technique to produce test cases through UML State diagram, instead of control flow graph, which is totally based on path oriented approach. UML State chart modeling is used to show the control and flow of data. It comprises of five following parameters.

- Initial state
- Final state
- No. of intermediate States
- Transition function
- Process function

Nirpal and Kale (2011[4]) presents use of genetic algorithms in software testing for automatically creation of test cases. In this method, the target path is selected and then sequences of operators of genetic algorithm are executed in order to achieve the good test cases. They have used triangle classification program for experiment and compare it with Young Chen method method [12]. But if the code consists of many functions, the probability of testing the function using above approach may not work. Fitness function is evaluated by calculating distance covered between the executed path and the target path. Doungsa et al. [23] used the genetic algorithm for generating the test cases in path testing from UML state machine. The test data covered the maximum transition of the possible path. The number of test data generated was increased in number as one test data test only one path, so for maximum coverage the author had increased the number of test cases. However the result is suitable for the simple program without loop.

Our work proposed considers the variable found in the code as a parameter as well as all of the below parameters and promises to give much better performance for testing the program code

Table 1: Parameters used in Software Testing (Source-Self)

Type	Author Name	Cross Over rate	Mutation Rate	Selection	Population	Iterations	Encoding
Path testing	Rijwan Khan, Mohd Amjad and Akhlesh Kumar Srivastava 2018	0.8	0.2	Random	20	10	Binary
Path testing	Bashir, M.B. and Nadeem, A., 2017	0.5	[1-0]	Random	50	10	Real

IIOAB JOURNAL
ISSN: 0976-3104

Mutation testing	Rijwan Khan, Mohd. Amjad	Condition: Mutation score > 0.5	Condition: Mutation score > 0.5	Tournament	_	20	Real
Data Flow	M. Boopathi1, R. Sujatha, C. Senthil Kumar, S. Narasimman	0.6-0.8	0.0-0.2	Random	5	5	Real
Path testing	Poonam Saini , Sanjay Tyagi	0.8	0.15	Roulette wheel	20	100	Binary
Path testing	Jasmine Minj Lekhraj Belchanden:2013	0.8	0.2	Roulette whell method	4	10	Binary
Path testing	Premal B. Nirpal and K. V. Kale(2011)	0.9	0.01	Random	1-1000	1000	Binary
Parallel Path testing	Doungsa, Dahal, Hossain & Suwannasart	0.5	0.05	Random	10*10	100	Binary

MATERIALS AND METHODS

Genetic Algorithm (GA) is a heuristic search process based on the survival of the fittest [1]. The algorithm was invented by Holland in 1960's [2]. The algorithm is very useful for finding the solutions of optimization problems. The search proceeds in the following way. The process starts with an initial population. The population is composed of chromosomes and each chromosome has cells. The cell may depict the inclusion or exclusion of a particular feature, in case of a Binary population. The number of chromosomes is initially, low. The process builds up a buffer of chromosome; each generation is more fit, on an average as compared to the previous generation [16]. This is achieved by the crossover and mutation operator. The crossover operator takes two chromosomes and produces a new chromosome. The operator can use one point mechanism or two point crossover or multipoint cross over, from amongst the many available options. In the one point cross over, one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent. In the case of two point cross over two points are selected and new chromosome(s) is produced. The mutation operator changes an existing chromosome and may end up producing a very good chromosome. As per the literature review the mutation operator inculcates diversity. Flip bit is one of the types of mutation. The above two operators are based on the selection of chromosomes. There are many types of selections like Rowlett wheel selection or tournament selection, to name a few. The Rowlett wheel selection selects a chromosome based on the commutative fitness. [17]

A chromosome is assigned a fitness based on the given problem. The fitness should contain the crux of the problem and can use functions like $f(x) = 1/(1 + e^{\lambda x})$, where λ can be found by empirical analysis. Note that the value of this function is between 0 and 1 and hence this function assigns a value which can be perceived as the probability of the chromosome being fit [18]. The termination of GA can be done either when the number of generations exceeds the set limit or there is practically no change in the chromosomes. The steps of Genetic Algorithm are depicted in [Fig. 1].

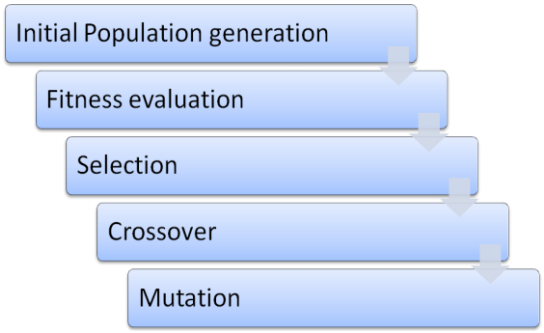

Fig. 1: Steps of each iteration of GA

The pivot concept behind the Genetic Algorithm based variable Importance evaluation (GABVIE) model is taking "variables" used in the program code as a constraint used to generate the test cases to test the path testing of a program code. The proposed algorithm assigns fitness to a variable depending upon its occurrence in the code and the frequency. The frequency of a variable is counted as follows. The occurrence of a variable in the 'if' block, assigns 1 to the frequency and that inside a 'for' loop (or 'while' for that matter) assigns n to it. The weight is determined by the frequency of the selected variables, multiplied by a constant λ, determined empirically. The above procedure would result in the selection of

IIOAB JOURNAL
ISSN: 0976-3104

the most important variables.

$$\text{Fittness} = 1/(1 + e^{\Sigma v_i w_i})$$

From amongst the given fitness function $1/(1 + e^{f})$ is chosen as it results in a value between 0 and 1, which can be perceived as the probability of the selection of variable. Here, v_i is 0 or 1, depending upon if the ith variable is selected or not and w_i is the weight. This is followed by the application of the GA process. The cross over and mutation applied till the population reaches its termination condition. The following algorithm and [Fig. 2] depicts the process.

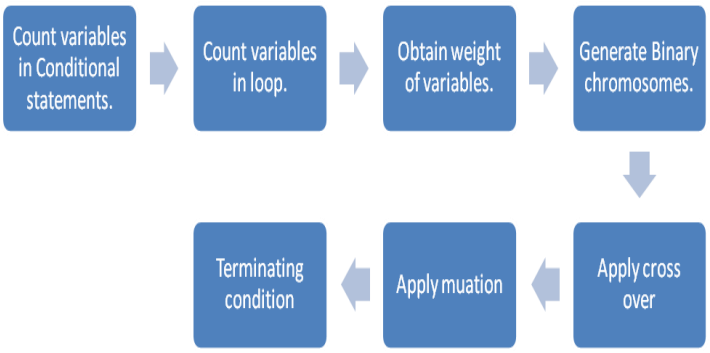

Fig. 2: Working model of GABVIE

Initially two arrays var and count are initialized to NULL. This is followed by populating the var[] array to variables in the program after scanning the program (Step 3). The program is again scanned for any occurrence of 'if' after which the count of the variable found is incremented by 1 and for 'loop' the count is incremented by 'm' the number of times, a loop runs. This is followed by the calculation of fitness. A binary population is then created. Row wheel selection is then applied to find the points on which one point cross-over is to be applied. Random mutation is then applied. The process is repeated till the stated numbers of generations are reached.

ALGORITHM:

Our key idea is to take a program code as a input and all the variables used in the program code are passed as parameters in the above model and cast this software testing problem as one of finding appropriate fitness of the functions in the program.

```
Algorithm 1: GABVIE v, t)
Input: Program statements with variables as parameters
Output: Fitness of the functions defined in program code
For i – 1 to n do
Scan
if equals (t, v[])
        then token <- var[ i++]
end if

Scan
If equals (token, block(if))
        Scan block(if)
        For i – 1 to n
                if equals (token, var [])
                count [] <- count[] +1
                end if
        end for
end if

if equals ( token, block(loop))
        Scan block(loop)
        For i – 1 to m
                if equals (token, var[])
                count <- count +m
                end if
        end for
end if

cell[]  <- var[]
```

IIOAB JOURNAL
ISSN: 0976-3104

```
create PopulationChromosome ( count, cell[])
for each cell
        if equals (cell [] , 1)
                calculate fitnessfunction
                fitness <-fit (count [])
                return fitness
        end if
end for

Select MostfittedChromosome
Point <-RowlettWheel (fitness, PopulationChromosome)
Calculate OnepointCrossover (point)
Calculate Mutation (random (PopulationChromosome))
Repeat until gen <=maxgen
```

The model has been implemented in Python and checked for few programs. It was observed that some conditions will be imposed on the input programs. In order to obtain the requisite fitness, the program has been divided in to two parts. The first part counts the occurrences of variables and decides the input to the fitness function and second part gives this input to the module implementing genetic algorithms.

[Table 2] shows the function used in the first in the first model and [Fig. 3] shows flow of the program.

Table 2: Functions used in model for Testing (Source-Self)

Function Name	Arguments	Description
_file	File Name	Opens code file and breaks it into a stream of tokens, returns the stream of tokens in a list named 'str'.
_var	str	Scans the input stream of code and selects the variables of data type int, char and float except for 'i' and 'j' which are reserved for loop iterations. Returns a list named 'var' containing the names of variables used.
_scan	str	Scans the code and detects the variables that are scanned by the user and takes the input by user. Returns a list 'scan_var' with the names of scanned variable and list 'value' with the respective value of those scanned variables.
_cou	temp, scan_var, value	'temp' contains the for loop condition; This function calculates the number of times a loop iterates. Returns the count of iterations evaluated by the expression in 'temp' as variable 'n'.
_count	str, var, scan_var, value,	Scans the code and calculates the number times a variable is used in the program. Returns a list 'count' containing the count of variables in 'var'

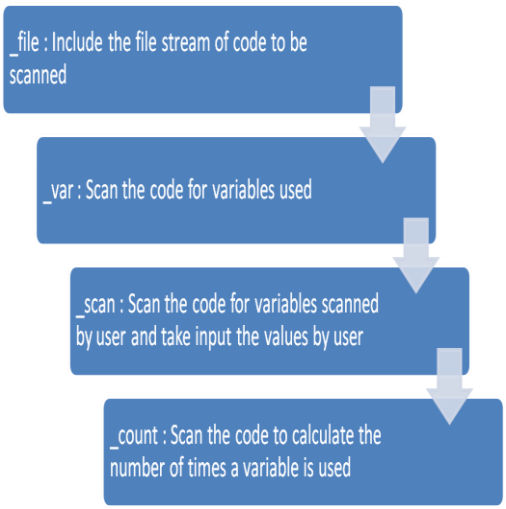

Fig. 3: Flow of Program

RESULTS

The model was tested the 10 C programs and it was found that the variables which were used more got the higher fitness value. It was also observed that the existing models cited above did not give requisite importance to the variables frequently used and hence was taking in the appropriate treatment of such variables leading to the test cases that may not contain such variables. In our case such variables were

IIOAB JOURNAL
ISSN: 0976-3104

being always a part of list being generated for creating test cases.

CONCLUSION

The work presented a novel model for selecting variables for testing. The model uses Genetic Algorithm. The fitness of the variables is determined using the frequency of occurrence. This is followed by the application of standard GA. The model has been carried out some results have been stated. The results are encouraging. Currently the model is being tested on a larger set of programs. The calculation of constants stated in the work would be done by extensive empirical analysis. The model has been proposed and the implementation being carried out. This model would pave way for the applicability of GA in testing.

ABOUT AUTHORS

Seema Sharma is a Research Scholar and Shaveta Bhatia is an Associate Professor, MRIIRS, Faculty of Computer Applications, Faridabad, India

CONFLICT OF INTEREST

There is no conflict of interest.

ACKNOWLEDGEMENTS

None

FINANCIAL DISCLOSURE

None

REFERENCES

[1] Alzabidi M, Kumar A, Shaligram AD. [2009] Automatic Software structural testing by using Evolutionary Algorithms for test data generations. International Journal of Computer Science and Network Security, 9(4):390-395.

[2] Girgis MR. [2005] Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm J UCS. 11(6): 898-915.

[3] Minj J, Belchanden L. [2013] Path Oriented Test Case Generation for UML State Diagram using Genetic Algorithm. International Journal of Computer Applications, 82(7).

[4] Nirpal PB, Kale KV. [2011] Comparison of software test data for automatic path coverage using genetic algorithm development, 4(5).

[5] Saini P, Tyagi S. [2012] Test data generation for basis path testing using genetic algorithm and clonal selection algorithm, Int J Sci Res, 3(6):2319-7064.

[6] Sharma A, Rishon P, Aggarwal A. [2016] Software testing using genetic algorithms Int J Comput Sci Eng Surv (IJCSES), 7(2):21-33.

[7] Srivastava PR, Kim TH. [2009] Application of genetic algorithm in software testing. International Journal of software Engineering and its Applications, 3(4):87-96.

[8] Goldberg DE. [1999] Genetic and evolutionary algorithms come of age. Communications of the ACM, 37(3):113-120.

[9] Chauhan N. [2010] Software Testing: Principles and Practices. Oxford university press.

[10] Bashir MB, Nadeem A. [2017] Improved genetic algorithm to reduce mutation testing cost, IEEE Access, 5:3657-3674.

[11] Boopathi M, Sujatha R, Kumar CS, Narasimman S. [2014] The mathematics of software testing using genetic algorithm. In Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization IEEE, 1-6.

[12] Chen Y, Zhong Y, Shi T, Liu J. [2009]. August Comparison of two fitness functions for GA-based path-oriented test data generation. In 2009 Fifth International Conference on Natural Computation IEEE, 4:177-181.

[13] Khan R, Amjad M. [2015], December, Automatic test case generation for unit software testing using genetic algorithm and mutation analysis. In 2015 IEEE UP Section Conference on Electrical Computer and Electronics (UPCON), IEEE, 1-5.

[14] Khan R, Amjad M, Srivastava AK. [2018] Optimization of Automatic Test Case Generation with Cuckoo Search and Genetic Algorithm Approaches. In Advances in Computer and Computational Sciences, Springer, Singapore, 413-423.

[15] Last M, Eyal S, Kandel A. [2005] Effective black-box testing with genetic algorithms. In Haifa Verification Conference Springer, Berlin, Heidelberg, 134-148.

[16] Mishra DB, Mishra R, Acharya AA, Das KN. [2019] Test Data Generation for Mutation Testing Using Genetic Algorithm. In Soft Computing for Problem Solving, Springer, Singapore, 857-867.

[17] De ABT, Martins E, De SFL. [2007] Generalized extremal optimization: an attractive alternative for test data generation. In Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM, 1138-1138.

[18] Knowles J, Corne, D. [1999] The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In Congress on Evolutionary Computation (CEC99), 1:98-105.

[19] Bhasin H, Singla, N. [2012] Harnessing cellular automata and genetic algorithms to solve travelling salesman problem. In International Conference on Information, Computing and Telecommunications, (ICICT-2012), 72-77.

[20] Blanco R, Tuya J, Adenso-Díaz B. [2009] Automated test data generation using a scatter search approach. Information and Software Technology, 51(4):708-720.

[21] Deb K, Pratap A, Agarwal S, Meyarivan TAMT. [2002] A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2):182-197.

[22] Engström E, Runeson P, Skoglund M. [2010] A systematic review on regression test selection techniques. Information and Software Technology, 52(1):14-30.

[23] Doungsa-ard C, Dahal KP, Hossain MA, Suwannasart T. [2007] An automatic test data generation from UML state diagram using genetic algorithm. In: Proceedings of the Second International Conference on Software Engineering Advances (ICSEA 2007). 25-31 Aug. 2007 Cap Esterel, France. IEEE Computer Society Press. 47-52.