Genetic Algorithm Approach For Test Case Generation Randomly: A Review

Deepak kumar[1], Manu Phogat[2],
[1]*Research Scholar, Dept. of computer science, GJUS&T, Hisar, India.*
[2]*Research Scholar, Dept. of computer science, GJUS&T, Hisar , India.*

Abstract. *The quality of software is dependent on testing as per user specifications and requirements. So it is quite challenging to design, prioritize and optimize test cases to achieve quality. Different testing tools can be used for software testing either manually or automatically. During the recent studies it is found that automated software testing is better than manual testing by using heuristic search. In this paper presents a survey on genetic algorithm approach for random generation of test cases in functional software testing*

Keywords — *put your keywords here, keywords are separated by comma.*

I. Introduction

Software testing is a process in which the runtime quality and quantity of a software is tested to maximum limits for a qualitative software. The software industry suffers with a heavy loss of $500 billion due to reduction in software quality. The software quality can be improved by performing automated testing and delivering the fully tested software, which can meet the user's specifications and requirements [1].Basically testing techniques are of two types functional testing and structural testing. Functional requirements are needed in functional testing, which is known as black box testing and structural testing is based on internal coding, known as white box testing. The combination of black box and white box testing is known as Gray box testing [2]. Software testing is the process of verification and validation to make sure that the software covers all the desired business and technical needs [3, 4].It is found that approximately 50% of the software Development resources are consumed during software testing. If the testing is performed using automated testing then it will lead to reduce in software Development cost by a significant margin [7, 8, 9].A test case consists of unique identifier, requirement references from a software specification, a series of steps, events, preconditions, input, output, expected result and actual result, which acts as input for the software testing [10, 11]. Over the course of time some software techniques which are applied successfully in the field of software testing are Genetic algorithm, neural networks, fuzzy logic, etc. The task of test case generation can be converted into an optimal problem by Meta heuristic search technique called as Genetic Algorithm [12, 13, 14]. Genetic Algorithms have been applied to many optimization problems in the field of software testing and this can be done by generating test plans automatically for functionality testing, as parallelism and search space operations are the important characteristics [11, 15, 16].This paper presents a survey of how GA is efficiently used to generate test cases for software testing. Further the paper is partitioned into 4 sections. Section 1 contains the Introduction, section 2 is a brief introduction to a GA, section 3 contains the related work in the field of test case generation using Genetic Algorithm for functional testing, and section 4, gives conclusion and future work.

An Introduction to Genetic Algorithm

Genetic algorithm is an evolutionary algorithm, which is developed by John Holland in 1970. It can be used to solve many complex and real life problems by producing high quality test data automatically [17, 18, 19]. It has emerged as a practical, robust optimization technique and search method and it is inspired by the way nature evolves species using natural selection of the fittest individuals. It is a best way to solve a set of problems with less information [11, 20]. The solution to a specific problem can be solved by a population of chromosomes, the strings of binary digits and each digit is called a gene and the population can be created randomly. Basically three different types of operators are used in the process of GA such as selection, crossover, and mutation [11, 18].

Selection: The use of selection operator is to select the best parents for performing other GA operations. Usually the selection is done on the basis of fitness value of the individuals, which is obtained from the fitness function. Fitness function can be defined as a specific function depending upon the criteria which returns a number indicating the acceptability of the program. This function is used in the selection process to determine the optimum point and the variants survive to the next iteration [21, 22]. Likelihood, close to Boundary Value and Branch Coverage are essential factors for a fitness function. Selection methods are of six different types such as roulette wheel, stochastic universal sampling, linear rank, exponential rank, binary tournament and truncation.

Crossover or Recombination: After selection, the crossover operation is applied to the selected chromosomes, which swaps genes or sequence of bits in the string between two individuals. For binary encoding different types of crossover operators are used like one point, two point, uniform and arithmetic. Cross over process is repeated with different parent individuals. Mutation is performed after crossover if the mutation probability is true for the given iteration.

Mutation: It is used to maintained genetic diversity in the population by altering chromosomes to introduce new good traits. Basically six types of mutation operators are used in Genetic algorithm such as Bit string, flip bit, boundary, uniform, non-uniform and Gaussian.

II. RELATED WORK

This section provides a short description about the different hybridized GA techniques applied for software testing. Mark Last et al. [14], proposed a hybrid fuzzy based GA to generate test cases for mutation testing. In this study a very minimal set of test cases were found. The faults in test cases are exposed by the use of mutated versions of the original method. The proposed method uses a Fuzzy Logic Controller (FLC) for obtaining the probability of crossover which differs according to the age intervals allocated during lifetime. The age and lifetime of chromosomes (parents) are defined by the FLC state variables. In their work an effective set of test cases are generated for a Boolean expression of 100 Boolean attributes by using three logical operators AND, OR, and NOT. An external application generates the correct expression randomly and one simple function is evaluated for each test case to generate an erroneous expression. Francisca Eanuelle et al. [20], has shown the generation of better test plans by using of GA method, for functionality testing. The method is applied in an unbiased manner to avoid the expert's interference. The fitness function used by them is shown in equation (1).

$$f_p = \sum_{i=1}^{k-1} t(l_i \rightarrow l_{i+1}) \qquad (1)$$

Where p = l1, l2,…,lk is a test plan or sequence of operations and t is a transition function for converting one operation li to the next operation li+1 in a sequence. The sequence is considered as better if the value of fitness function is high. Ruilian Zhao et al. [24], used the GA and neural network for the functional testing of the software under test, and they applied the improved Genetic algorithm to the function model, created using neural network. The following fitness function, shown in equation (2) is used.

$$f = \begin{cases} \dfrac{1}{|c-g|}, & c \neq g \\ f_{max}, & |c-g| \leq 10^{-8} \end{cases} \qquad (2)$$

Here c represents the actual output and g represents the goal output of the software under testing. If the fitness value of the proposed algorithm reaches or crosses the maximum value of possible outcome, then the algorithm terminates its execution and the current individual is termed as the best test inputs for the corresponding outputs. The authors found that proposed GA can generate better test cases with high efficiency.
Li, Zhang and Kou [25], applied the GA to get the local optimal solution to a specific problem to achieve improved performance They applied a new algorithm called as Genetic-Particle Swarm Mixed Algorithm (GPSMA) to automatically generate software test data. The proposed technique uses the update mode in each individual to replace the mutation process in the algorithm which is based on population division. The proposed algorithm can generate and search specific test data in a domain to satisfy the test condition.

Xuan Peng et al. [15], proposed an approach US-RDG, for web application in terms of gray box testing, by combining User Session data with Request Dependence Graph (RDG) to generate test cases automatically. Their simulated results shows that US-RDG effects better than the traditional user session-based testing by higher path coverage and fault detection rate for a small size of test suite. They used the conception as transition relation in the form of "page → request → page". Transition relations are used to indicate the relationship between pages and requests from structural point transition relations in the specific application can be extracted from structural analysis by RDG and finally a GA heuristic was proposed to generate test cases to cover maximum of transition relations by mixing different user sessions. They found the performance of US-RDG is very well in test case generation for web application. They used the fitness function of a chromosome as shown in equation (3).

$$\textbf{Fitness} = (\alpha * |\textbf{CDTR}| + |\textbf{CLTR}|) / (\alpha * |\textbf{DTR}| + |\textbf{LTR}|) \qquad (3)$$

Where |CDTR| and |CLTR| denotes the number of data and link dependence transition relations covered in the chromosome. The fitness value achieves 1 when a chromosome covers all the data and link dependence in transition relations present in the specific application. The authors introduced a parameter α, indicating the coefficient of the data dependence transition relation. They have taken α as

much as greater than 1 to increase the proportion of the data dependence transition relation and assigned 1 to the coefficient of the link dependence in transition relation. The chromosomes have a bigger fitness which covers more data dependence transition relations.
Ali Shahbazi et al. [5], used a multi objective optimization in black box string test case generation for random testing and adaptive random testing. The authors examined many string distance functions and hence they introduce two objectives for effective string test cases such as the length distribution of the string test cases and the diversity control of the test cases within a test set. They used one diversity-based fitness function to generate optimized test sets to reveal faults more effectively and it is shown in equation (4).

$$F_D = \sum_{i=1}^{test\ set\ size} dist\left(t_i, \beta\left(t_i, test\ set\right)\right) \quad (4)$$

In the above function ti denotes the ith test case and β is its nearest test case in the test set and the summation is performed between the two distances test cases. They found the higher value of the fitness function results, the more diverse distribution of test cases. After an extensive study of different testing techniques, we came to learn GA parameter is efficiently used for generating test cases in functional software testing.

III. CONCLUSION AND FUTURE WORK

In this paper, authors studied how different types of hybridized Genetic Algorithms are helped in efficient software testing with increasing number of test case generation. It provides a means of an automatic test case generator. Genetic algorithm can also be used with the neural networks and fuzzy systems for performing different types of testing to improve the performance.

In future a new hybridized algorithm can be designed by taking a better fitness function which will help us to evaluate the efficiency of test cases and further to increase the efficiency of the test result by changing the input parameters, by increasing the number of generations and obtain values for different number of population. It is also planned to use hybridized GA with other soft computing techniques such as neural network for optimized test case generation in web-based application software.

REFERENCES

[1] Jones,C., Bonsignour, O.: The Economics of Software Quality, Pearson Education Inc., 2012

[2] Shivani, A., and Pandya. V., Bridge between Black Box and White Box–Gray Box Testing Technique, International Journal of Electronics and Computer Science Engineering 2.1 (2012): 175-185.

[3] Chauhan, N., Software Testing: Principles and Practices, Oxford University Press, 2010.

[4] Jogersen, P. C., Software testing: A craftsman approach, 3rd edition, CRC presses, 2008.

[5] Shahbazi, A., and Miller, J., Black-Box String Test Case Generation Through A Multi-Objective Optimization, IEEE Transactions On Software Engineering, 42(4), 2016

[6] Michael, C.C , McGraw, G. E., Schatz, M. A. and Walton, C.C., Genetic Algorithms for Dynamic Test Data Generation, Proceedings of the 1997 International Conference on Automated Software Engineering (ASE'97) (formerly: KBSE) 1997 .

[7] Doungsaard, C., Dahal, K., Hossain, A., Suwannasart, T., Test data generation from UML state machine diagrams using GAs, International Conference on Software Engineering Advances (ICSEA 2007). IEEE, 2007.

[8] Srivastava, P.P., and Kim, T., Application of genetic algorithm in software testing, International Journal of software Engineering and its Applications, Vol.3, No.4, pp.87 – 96, 2009.

[9] Berndt, D.J., and Watkins, A., High volume software testing using genetic algorithms, Proceedings of the 38th Annual Hawaii International Conference on System Sciences Washington, DC, USA: IEEE Computer Society, Vol. 9, pp. 318–326, 2005.

[10] Dixit, S., and Tomar, P., Automated test data generation using computational intelligence, Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), 4th International Conference on. IEEE, 2015.

[11] Sharma, A., Patani , R. ,and Aggarwal, A., Software Testing using Genetic Algorithms, International Journal of Computer Science & Engineering Survey Vol.7, No.2, 2016

[12] Moataz A. A, and Ali, F., Multiple-path testing for cross site scripting using genetic algorithms, Journal of Systems Architecture Vol.64, pp.50-62, 2016.

[13] Yang, S., Man, T., Xu, J., Zeng, F., Li, K., RGA: A lightweight and effective regeneration genetic algorithm for coverage-oriented software test data generation, Information and Software Technology, Vol.76, pp.19-30, 2016.

[14] Last, M., Eyal, S., Effective black-box testing with genetic algorithms, Lecture notes in computer science, Springer,pp. 134 -148, 2006.

[15] Peng, X., & Lu, L., A new approach for session-based test case generation by GA. In Communication Software and Networks (ICCSN),IEEE 3rd International Conference on IEEE, pp. 91-96, 2011.

[16] Zhao, R., lv, S., Neural network based test cases generation using genetic algorithm, 13th IEEE international symposium on Pacific Rim dependable computing. IEEE, pp.97 – 100, 2007.

[17] Srivastava, P.R., and Kim, T.H ., Application of genetic algorithm in software testing, International Journal of software Engineering and its Applications, Vol.3, No.4, pp.87 – 96, 2009.

[18] Riberio, J.C.B., Zenha-Rela, M.A., De vega, F.F., A strategy for evaluating feasible and unfeasible test cases for the evolutionary testing of object oriented software", AST' 08. ACM, 2008.

[19] Goldberg, D.E., Genetic Algorithms: In search, optimization and machine learning, Addison Wesley, M.A, 1989.

[20] Wappler, S., Lammermann, F., Using evolutionary algorithms for unit testing of object oriented software, GECCO. ACM, pp.1925 – 1932, 2005.

[21] Vieria, F. E., Martins, F., Silva, R., Menezes, R., Braga., M., Using Genetic algorithms for test plans for functional testing, 44th ACM SE proceeding, pp.140 – 145,2006.

[22] Mathur, A.P., Foundation of Software Testing, 1st edition Pearson Education 2008.

[23] Rauf, A., Anwar, S., Jaffer, M. A., & Shahid, A. A., Automated GUI test coverage analysis using GA, In Information Technology: New Generations (ITNG), Seventh International Conference , IEEE, pp. 1057-1062, 2010.

[24] Andalib, A., and Babamir, S.M., A New Approach for Test Case Generation by Discrete Particle Swarm Optimization Algorithm, The 22nd Iranian Conference on Electrical Engineering, 2014.

[25] Zhao, R., & Lv, S., Neural-network based test cases generation using genetic algorithm. In Dependable Computing, 13th Pacific Rim International Symposium IEEE, pp. 97-100, 2007

[26] Li, K., Zhang, Z., Kou, J., Breeding Software Data with Genetic Particle Swarm Mixed Algorithm, Journal of Computers, Vol.5, No.2,pp. 074-085, 2010.