Reducing Testing Cost using Stochastic evolutionary Technique

Ankita Vashisth[1], Marender Singh Dagar[2]
[1]*M.Tech (CS), SRCEM Palwal, INDIA*
[2]*Asst.Prof, SRCEM Palwal,INDIA*

***Abstract*- Software testing is an inevitable activity in software development. It is a critical determinant of software quality and consumes approximately 50% of software development costs. Test case generation is a vital component of software testing and greatly influences the efficiency and effectiveness of any software test hence; it has been extensively studied and is regarded as an important subject area in software testing. Any guarantee of high software quality requires maximum test adequacy coverage using test cases during software testing. This paper presents a comparative study of the methods used for the automatic generation of test cases during software testing and explores the limitations of each method.**

***Index Terms*- Software testing; Test case generation; automatic test case generation methods**

I. INTRODUCTION

Software testing is a necessary and an integral section of software engineering development [1]. However, testing is an intensive work and costly. It is often account greater than 50% of total cost of the development. Therefore, it is important to decrease the cost and improve the software testing effectiveness by automate the process of testing [2]. Among the different testing activities, test case generation is one of the most mentally overwork and most critical, because it can have a powerful effect on the effectiveness and efficiency of total testing process [3][4]. It is not amazing that most of researches effort in the last decades has been expend on the automatic test case generation.

A perfect set of test cases is one that has high chance of discovering the previous unknown errors and a successful test run, which discovers these errors. To uncover all potential errors in program, detailed testing is required to examine all possible input and logical execution paths but it is neither possible nor economically feasible. Thus, the actual goal for software testing is to increase the finding errors probability using a limited number of test cases that perform in less time with less effort [5].

Various metrics have appeared, and applied, to evaluate the test cases generated quality like the cost, time, effort, and generation complexity as well as coverage criteria. Optimizing or even improving test cases quality can be intend of several researchers [6][7][8]. It can take many forms, like minimizing time or effort testing, minimizing the complexity or the generation algorithms cost, maximizing the coverage function as well as another reliability and quality matters. Also decreasing the test cases or test data generation can be an optimization form[9].

A test adequacy criterion provides a measurement of test suite quality and can be used to guide test generation. There are three widely applied kinds of coverage criteria namely mutation coverage (which evaluates the fault- revealing capability of a test suite) code coverage (which describes the extent to which source code program has been examined) and specification based coverage (which specify the percentage of testing requirements identified in a specification that have been covered by the test suite). Code coverage has branches which includes branch coverage, statement coverage and path coverage while specification based coverage includes types like requirements coverage, test data adequacy, boundary value analysis [10].

The present test case generation methods can be categorized into black-box testing and white-box testing depends on type of testing. Black-box test cases are specified from the description of the software under test [11]. White-box test cases are obtained from the inner software structure [12]. However, in both the cases it is difficult to achieve complete automation of the test case design [13].

This paper discusses an overview of different approaches that is used in generated test cases automatically which is the critical part in software testing process and the types of coverage that is used in these methods.

This comparative evaluation study helps the researchers to choose the suitable method that generate appropriate test cases with minimum test suite size and maximum coverage criteria as well as in minimum execution time. We described how to evaluate generated test cases, and introduce a classification of evaluation approaches.

II. RELATED WORK

Several algorithms based on genetic algorithm [14,15] and swarm intelligence [16,17] ie.ant colony optimizations and bee colony optimizations have been proposed for test case selection and prioritization from a large test suite. Sthamer[18] and Pargas et al [19] applied GA for automatic testdata generation in his thesis. A Strategy for using GA to automate branch and fault-based Testing [20] and automatic structural testing using genetic algorithms [21] is done by Jones et al. Lin and Yeh worked on GA for automatic test data generation based on path based testing [22]. An evolutionary approach is developed to dynamic test data generation by Anastasis and Andreas [23]. Harman et al proposed an approach to reduce the input domain using search based technique [24]. In fact, the genetic algorithm is also used to generate test data automatically [25].A lot of work is done by researchers on optimization of test cases. Mala et al has developed a hybrid genetic algorithm based approach for quality improvement and optimization of test cases[26] and Eric et al analyzed the effect of fault detection of test set when its size is minimized [27]. The concept of Artificial Bee Colony algorithm was introduced by Karaboga [28,29]. Chong et al [30] applied honey bees foraging behavior model to the job scheduling problem. McCaffrey et al [31] generates pair wise test sets using a simulated bee colony algorithm. Mala et al [32] presented a new, non pheromonen based test suite optimization approach inspired by the behavior of biological bees. Dahiya et al [33] presented an ABC algorithm based approach for automatic generation of structural software tests.

III. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and positions with following equations[34]:

v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) ……. (a)

present[] = persent[] + v[] ……..(b)

v[] is the particle velocity, persent[] is the current particle (solution). pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. Usually c1 = c2 = 2.

IV. PROPOSED WORK

In this paper, we proposed a new approach to reduce the cost of testing by test case suite reduction. The proposed technique is based on concepts of Swarm Intelligence. The technique selects the set of test case from the available test suite that will cover all the faults detected earlier in minimum execution time. Here particles are used as agents who explore the minimum set of test cases. The particles start flying from their current position following the current optimal path. After each iteration, Each particle updates its velocity and position. This updation is done according to the two optimal values attained by some particle. The process is repeated till any of the particle has discovered a set of test cases that covers nearly all faults detection. The prerequisite for the proposed algorithm is a test suite 'T' of 'n' test cases. The result is subset 'S', which consists of m test cases(m<=n),such that the test cases are selected on the basis of maximum fault coverage capacity in minimum execution time.

The assumptions taken for the proposed algorithm is as follows:

- Given the original test suite, T={t1,t2……tn}.
- Set of all faults, F={f1,f2,…….fk}.
- Each test case {t1,t2,…tn} in the original test suite covers some or all the faults from 'F'.
- Each test case will be represented in binary form. Each test case is of 'k' bits (k is the total number of faults).Each bit of the test case depends upon the capacity of detecting that fault. Starting from the leftmost bit, the bit is 1 if it detects Fault fk else 0 and so on.
- Number of particles to search through the test case space is n (number of test cases).

For each particle
   Initialize particle
END

Do
   For each particle
      Calculate fitness value

If the fitness value is better than the best fitness value (pBest) in history
set current value as the new pBest
End

Choose the particle with the best fitness value of all the particles as the gBest
For each particle
Calculate particle velocity according equation (a)
Update particle position according equation (b)
End
While maximum iterations or minimum error criteria is not attained

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T1		X			X			X	X	
T2	X		X			X	X			X
T3		X		X			X	X		X
T4			X			X			X	
T5	X			X	X					
T6			X					X		X
T7				X			X			
T8	X				X			X		X
T9		X	X			X			X	
T10	X			X			X			X

Table1: Test Case and Fault Coverage

Test Case	Binary Form
T1	0100100110
T2	1010011001
T3	0101001101
T4	0010010010
T5	1001100000
T6	0010000101
T7	0001001000
T8	1000100101
T9	0110010010
T10	1001001001

Table2: Binary Representation of Test Cases

V. RESULT ANALYSIS

The technique was implemented using matlab tool.
Table 3 shows the reduced number of test cases and Test 4 shows their fault coverage as well.

Test Case	Binary Form
T1	0010111101
T2	1001101110
T3	1111111010
T4	1101101011

Table 3:Reduced Test Cases

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T1			X		X	X	X	X		X
T2	X			X	X		X	X	X	
T3	X	X	X	X	X	X	X		X	
T4	X	X		X	X		X		X	X

Table 4:Reduced Test Cases Fault Coverage

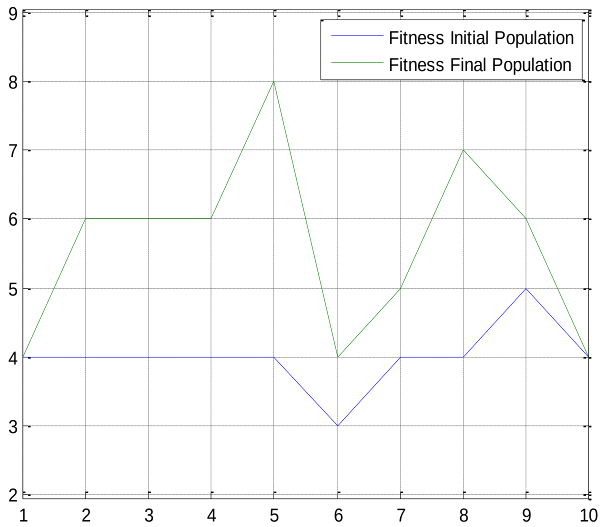

Figure: Initial v/s Final Fitness

VI. CONCLUSION AND FUTURE SCOPE

We have proposed test case selection approach from a large test suite using technique based on Particle Swarm Optimizations. The technique was implemented and tested for a sample data of 10 test cases. The technique developed using this approach was able to identify and reduce the test data. The reduced test cases were having higher fault coverage.

Issues of future research include automation of the technique and applying it on large and complex software. We also aim to compare it to ant colony optimizations algorithms and genetic algorithms.

REFERENCES

[1] Young, M. (2008). Software testing and analysis: process, principles, and techniques.John Wiley & Sons.

[2] Anand, S., Burke, E., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., & Zhu, H. (2013). An Orchestrated Survey on Automated Software Test Case Generation. Journal of Systems and Software.

[3] Bertolino, A. (2007, May). Software testing research: Achievements, challenges, dreams.
In Future of Software Engineering, 2007. FOSE'07 (pp. 85-103). IEEE.

[4] Pezz`e, M. and Young, M., 2007. Software Testing and Analysis - Process, Principles and
Techniques. Wiley.

[5] Devasena, M. G., & Valarmathi, M. L. (2012). Search based Software Testing Technique for Structural Test Case Generation. International Journal of Applied Information Systems (IJAIS), 1(6).

[6] Farooq, U., & Lam, C. P. (2009, April). Evolving the Quality of a Model Based Test Suite.In Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on (pp. 141-149). IEEE. *1st Technology, Education, and Science International Conference (TESIC) 2013*73

[7] Kosindrdecha, N., & Daengdej, J. (2010). A Black-Box Test Case Generation Method.
International Journal of Computer Science and Information Security (IJCSIS).

[8] Harman, M., Kim, S. G., Lakhotia, K., McMinn, P., & Yoo, S. (2010, April). Optimizing for the number of tests generated in search based test data generation with an application to the oracle cost problem. In Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on (pp. 182-191). IEEE.

[9] Boghdady, P. N., Badr, N., Hashem, M., & Tolba, M. F. (2011). Test Case Generation and Test Data Extraction Techniques. Inter. J. Electr. Comput. Sci, 11(3), 87-94.

[10] ZHENG, W. (2011). Automatic Software Testing Via Mining Software Data, PhD thesis, The Chinese University of Hong Kong.

[11] Parnami, S., Sharma, K. S., & Chande, S. V. (2012). A Survey on Generation of Test Cases and Test Data Using Artificial Intelligence Techniques , UACEE International Journal of Advances in Computer Networks and its Security, pp. 16-18.

[12] Singh, K., & Kumar, R. (2010). Optimization of Functional Testing using Genetic Algorithms. International Journal of Innovation, Management and Technology, 1(1), 2010-0248.

[13] Geetha Devasena, M. S., & Valarmathi, M. L. (2012).Meta Heuristic Search Technique for Dynamic Test Case Generation. International Journal of Computer Applications. 39 (12)

[14] M.J.Harrold, R.Gupta, and M.L. Soffa,“ A methodology for controlling the size of the test suite, ” ACM Transaction on Software Engineering and Methodology, pages 270-285, July 1993.

[15] H.Agrawal ,J.R. Horgan, and E.W. , Krauser, “Incremental regression testing,” In: Proc.
Conference on Software Maintenance, pages 348-357,1993.

[16] R.Bahsoon, N. Mansour, “Methods and metrics for selective regression testing,” In Computer Systems and Applications, ACS/IEEE International Conference, pages 463-465, 2001.

[17] G. Rothermel, R.H. Untch, C. Chu, and M.J.Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no.10, pages 929-948, Oct. 2001.

[18] S.Elbaum, Alexey G. Malishevsky, andG.Rothermel, "Test case prioritization: A family of empirical studies," IEEE Transactions on Software Engineering, vol. 28, NO.2, pages 159-182, Feb.2002.

[19] K. K. Aggrawal, Y. Singh, A. Kaur, " Code coverage based technique for prioritizing test cases for regression testing ," ACM SIGSOFT Software Engineering Notes , vol 29 Issue 5 September 2004.

[20] J.Holland, "Adaption in Natural and Artificial Systems", Ann Arbor, MI: University of Michigan Press,1975.

[21] D. Goldberg, "Genetic Algorithms in Search Optimization and Machine Learning", New York,Addision Wesely, 1989.

[22] Wikipedia; http://en.wikipedia.org/wiki/Swarm_intelligence.

[23] Scholarpedia;http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm.

[24] H.H. Sthamer, "The automatic generation of software test data using genetic algorithms, Ph.D thesis, University of Glamorgan 1996.

[25] R.P Paragas, M. Harrolg and R.Peck , "Test data generations using genetic algorithms", Software testing verification and reliability, vol.9, no4, pp263-282,1999.

[26] B .Jones, D.Eyres and H .Sthamer ,"A strategy for using genetic algorithms to automate branch and fault based testing", the computer journal ,vol 41, no.2pp. 98-107,1998.

[27] B.F Jones, H.H Sthamer and D.Eyres,"Automatic structural testing using genetic algorithms", Software engineering journal,vol.11,no.5,pp.229-306, 1996.

[28] J.C. Lin, P.L. Yeh," Automatic test data generation for path testing using Gas", Department of Computer Science and Engineering, Tatung University,1999.

[29] A.Anastasis, A. S. Andreou," Automatic, evolutionary test data generation for dynamic software testing", Journal of Systems andSoftware Volume 81, Issue 11, Pages 1883-1898, November 2008.

[30] M.Harman,Y.Hassoun,K.Lakhotia,P.McMinn,J.Wegener," The impact of input domain reduction on search-based test data generation",in the proceedings of ACM SIGSOFT, ISBN: 978-1-59593-811-4,2007.

[31] Marc Roper, Iain Maclean, Andrew Brooks,James Miller and Murray Wood. Genetic Algorithms and the Automatic Generation of Test data,1995.

[32] D.J.Mala , V.Mohan, "Quality Improvement and Optimization of Test Cases-A Hybrid Genetic Algorithm Based Approach", ACM SIGSOFT ,May 2010.

[33] W.W.Eric, ,R.H.Joseph, L.Saul and Aditya P.Mathur,"Effect of Test Case Minimization of Fault Detection Effectiveness",Software pPractice and Experience,Vol.28,No.4, pp. 347-369, 1998.

[34] http://www.swarmintelligence.org/tutorials.php