A Review of use of Evolutionary Techniques in reducing Test Cases

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 prioritization involves scheduling test cases in an order that increases the effectiveness in achieving some performance goals. One of the most important performance goals is the rate of fault detection. Test cases should run in an order that increases the possibility of fault detection and also that detects the most severe faults at the earliest in its testing life cycle. Regression Testing is an usual and a very costly activity to be performed, often in a time and resource constrained environment. Thus we use techniques like Test Case Selection and Prioritization, to select and prioritize a subset from the complete test suite, fulfilling some chosen criteria. Present paper gives the approach into existing single objective test cases prioritization and optimization using techniques such as Genetic Algorithms, Ant Colony Optimization. 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

As any software system is developed changes are made to the software. Changes are done to introduce new features and functionalities. So after upgradation it is necessary to test the software to make sure that the system is working as intended. Hence, during regression testing the new test cases along with the old ones are executed to certify the functionality of the software. So, it becomes a tough task to carry out regression testing as size of test suite grows.[1]

In order to assist the software engineer in regression testing, test suite minimization techniques can be used

Test Suite Minimization Approach: Initially, a test suite T is given with all the possible test cases to test the software completely. Then use some algorithm to reduce T to get the test suite reduction T '.

T ' is not redundant, meaning that if any of the test cases is removed from T ' , the rest of the test case does not meet all the requirements.[3]

Test suite can be optimized based on fault detection, execution time and coverage.

The NP-completeness nature of test suite minimization problem inspired many researchers to experiment with different heuristics for its solution.

In recent years, biological intelligent heuristic optimization algorithms have become one of the mainstream methods to solve the non-linear, non-

differential, multi-peak and complex problems. Many different bionic algorithms have been introduced by scholars from different countries, inspired from the foraging behaviors in the nature creature. Dorigo M et al. proposed the *Ant Colony Optimization (ACO)* [4] in 1991; Eberhart and Kennedy proposed the *Particle Swarm Optimization (PSO)* [5] in 1995; Passino et al proposed the

Bacterial Foraging Optimization (BFO) [6] in 2002. Because of the advantages of parallel searching, jumping out of local minimum easily and so on, BFO is becoming a hot spot of bionic algorithm. Since the researching work of this algorithm in China is at the beginning stage and the randomness of bacterial chemotaxis in the algorithm, it leads to the slow chemotaxis speed and inefficient. So that, combining some common mechanisms

and principles of intelligent bionic algorithm with the differences in the internal operation mechanism becomes a natural way to optimize the algorithm.[7]

II. OPTIMIZATION TECHNIQUES

The optimization algorithms are explained below :

A. Genetic Algorithm

Genetic Algorithms are population-based general purpose algorithms used to find accurate or estimated solutions to optimization and search problem.

They are stochastic search techniques based on the phenomenon of natural selection and genetics.GA begins with an initial population which is a random set of solutions. Each individual solution in the population is called a Chromosome. A chromosome can be a binary digit or any other data structure. The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated, using some measure of fitness.

Selection, Crossover and Mutation are three basic operators responsible for GA and these are described below:

1. Selection : A new generation is formed by selecting those chromosomes that satisfy the fitness value criteria. Suitable chromosomes with higher probability are selected. Some parents and offsprings are retained while others are rejected so as to keep the population size constant. After several generations the algorithm converge to optimal or near optimal solution.

2. Crossover : the exchange of parents' information produces an offspring.

3. Mutation : Randomly change one or more digits in the string representing an individual.

B. *Particle Swarm Optimization*

Particle Swarm Optimization (PSO) is a population-based meta-heuristic algorithm developed from the simulation of social models of bird flocking, fish schooling, and swarming able to find best possible solution(s) to the non-linear numeric problems. PSO was first introduced in 1995 by Eberhart and Kennedy. However, PSO can easily be trapped in local optimal point when dealing with some complex and multimodal functions.

PSO involves a number of particles, which are initialized randomly in the space of the design variables. These particles fly through the search space and their positions are updated based on the best positions of individual particles and the best position among all particles in the search space which in truss sizing problems corresponds to a particle with the smallest weight[5]. In PSO, a swarm consists of N particles moving around in a D-dimensional search space. The position of the jth particle at the kth iteration is used to evaluate the quality of the particle and represents candidate solution(s) for the search or optimization problems. PSO's exploration ability, the inertia weight is now modified during the optimization process.

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.

Unlike GA there are no selection, crossover and variation operation in PSO. So its algorithm is very simple and has a high execution speed.

However if some particle finds a present optimal point then other particle will be closed to it rapidly. Hence the diversity of whole swarm and its global searching ability will be weakened obviously. [17]

C. *Ant Colony Optimization*

Ant Colony Optimization(ACO) algorithm was proposed by Marico Dorigo in 2005.[8] ACO is a probabilistic technique for solving computational problems which can be used for searching shortest paths.[9]

ACO deals with two important processes, namely: Pheromone deposition and trail pheromone evaporation. Pheromone deposition is the phenomenon of ants adding the pheromone on all paths they follow. Pheromone trail evaporation means decreasing the amount of pheromone deposited on every path with respect to time. Updating the trail is performed when ants either complete their search or get the shortest path to reach the food source.[16]

Basic Principle of the algorithm is that Ants always find a shortest path between the nest to food source, which mainly depends on a hormone-pheromones. The shorter path contains more pheromone, the probability of choosing that path by ants is greater and finally ants colony will find a shortest path.[3]

ACO technique has been already used in solving various combinatorial problem such as knapsack problem, travelling salesman problem, distributed network, telecommunication network, vehicle routing, test data generation.[9]

Though ACO is next generation technique for optimization problems but it is not providing good solutions of problems like multiple objectives optimization, Dynamic Optimization Problems, the Stochastic Optimization Problems, continuous optimization and Parallel Implementations of the constraints.[10]

Most ant colony optimization algorithms use this algorithm demonstrated below :[11]

```
Initiation of the parameters which determines the
                pheromone trail

While (until result conditions supplied)
      do Generate Solutions

         Apply Local Search
         Update Pheromone Trail
End
```

D. BFO Algorithm

Bacteria Foraging Optimization Algorithm (BFOA), given by Passino in 2002 , belongs to nature-inspired optimization algorithms. Main idea behind the algorithm is group foraging strategy of E.Coli. bacteria in order to maximize energy obtained per unit time. Communication also occurs between individual bacterium to improve the searching strategy. This algorithm consists of four prime steps[13] :

1. Chemotaxis**:** Here, swimming and tumbling are the two prime ways which define the manner in which bacteria search for food. Swimming means moving in a pre-specified direction. Tumbling means moving in a completely new direction. Mathematically, tumble of any bacterium can be given by multiplication of $\phi(j)$ and $C(i)$,where $\phi(j)$ is unit length in random direction and $C(i)$ is step length. In case of swimming, $C(i)$ is constant.

2. Swarming: For the algorithm to converge at the optimal solution, it is required that the optimum bacteria attract other bacteria so that together they converge at the solution point quickly. To achieve this, a penalty function is added to the original cost function on the basis of relative distances of each bacterium from the fittest one. Penalty function becomes zero when all the bacteria have reached to the solution point.

3. Reproduction: Here, the fittest bacteria are divided into two groups. The weaker set of bacteria are replaced by other more fit set of bacteria. This keeps the population of bacteria constant throughout the evolution process.

Elimination and Dispersal: Because of changes in environment some bacteria may be killed or may be dispersed to a new place. In BFOA, this phenomenon is simulated by liquidating some bacteria and initialing new replacements randomly in the search space. It helps in reducing the probability of being trapped in pre-mature solution point.[12]

III. RELATED WORK

Several algorithms based on genetic algorithm [14,15] and swarm intelligennce [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.

Sangeeta Sabharwal et. al n this paper a GA based approach is proposed for identifying the test path that must be tested first. Test paths or scenarios are derived from activity diagram and state chart diagram respectively. The proposed approach makes use of IF model and GA to find the path to be tested first.

S. Raju and G. V. Uma in this paper the regression testing based test suite prioritization technique is illustrated. A new prioritization technique is proposed for requirement based System level test cases to improve the rate of fault detection of severe faults.

Chartchai Doungsa et al, n this paper author proposed an approach for generating test data from UML state diagram using genetic algorithm.This approach helps software developers to reduce their effort in generating test data before coding.

Arvinder Kaur and Shubhra Goyal in this paper a new Genetic Algorithm to prioritize the regression test suite is introduced that prioritize test cases on the basis of complete code coverage. The genetic algorithm would also automate the process of test case prioritization. The results representing the effectiveness of algorithms are presented with the help of an Average Percentage of Code Covered (APCC) metric.

Bharti Suri and Shweta Singhal in this paper presents an implementation of an already introduced Ant Colony Optimization Algorithm for Test Case Selection and Prioritization. Graph representation and example runs explained in the paper show how the random nature of ACO helps to explore the possible paths and choose the optimal from them. Results show that ACO leads to solutions that are in close proximity with optimal solutions. In this study a tool ACO_TCSP for the same has been developed and applied on an example. Though in these tests the best solution was not found for all cases still the results obtained are in close proximity to the optimal results. The reduction in test suite size is achieved to be 62.5% in all the 4 test runs

IV.CONCLUSION & FUTURE WORK

Test data generation is one of the key issues in software testing. A properly generated test suite may not only locate the errors in a software system, but also help in reducing the high cost, efforts associated with software testing. Present work surveyed various techniques of software test case optimization. First we summarized traditional and advanced test optimization techniques, and then we identified gaps in existing techniques. Optimization of test cases is multi-objective optimization, NP complete and peculiar nature problem. Soft computing can be used for these type problems. whose inexact solutions driving is computationally hard tasks such as the solution of “NP-complete problems”.

In conclusion, a lot of test cases optimization techniques have been developed for achieving software testing effectiveness and fault coverage. Review of existing literatures has identified that there are several objectives of test case optimization like maximum number of defect detecting capability, minimum test design efforts/cost, minimum execution cost, maximum coveragebility of client requirements & codes, maximum mutant killing score and so forth. Therefore optimization of test cases should be treated as multi-objective optimization problem. However most of test cases optimization approaches are single objective. Single objective formulation of test cases optimization problem is not justified and not meeting the objectives of testing. Some objectives are conflicting in nature, coveragebility of one objective will suffer other objective while considering all objectives concurrently. So, there is strong need to shift the paradigm from single objective test case optimization to multi-objective test case optimization. Moreover for these techniques, soft computing approaches like Genetic Algorithms, Fuzzy Logic, Artificial Neural Network etc may be well suited for experimentation and validation purpose

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