Integrated Pairwise Testing based Genetic Algorithm for Test Optimization

Baswaraju Swathi[1]
Research Scholar
Department of CSE
Jain University, Bengaluru
India

Dr.Harshvardhan Tiwari[2]
Associate Professor, Centre for Incubation
Innovation, Research and Consultancy (CIIRC)
Jyothy Institute of Technology, Bengaluru
Karnataka, India

***Abstract*—Generation of Test cases in software testing is an important and a complex activity as it deals with diversified range of inputs. Fundamentally, test case generation is considered to be a multi-objective problem as it aims to cover many targets. Deriving test cases for the Web Applications has become critical to the most of the enterprises. In this paper, a solution for generating test cases for web applications is proposed; the solution uses the System Graph (consisting of links and data dependencies) considering that test cases were based on a combination of input values and data dependencies. Pairwise testing is used to derive the test cases to be executing from entire test cases and then a genetic algorithm is proposed to generate test cases specific to functional testing. The proposed approach was tested through two distinct experiments by measuring the code coverage at every generation and results show that genetic algorithm used increased the fitness value and code coverage. Overall, the results of the paper validate the proposed approach and algorithm, having potential in further construct an automated integrated solution for generating test cases for the entire process.**

Keywords*—Test case generation; genetic algorithm; multi objective optimization; pairwise testing; test optimization; fitness value

I. Introduction

The key point regarding the usage of Soft Computing in testing is towards maximizing the quality of software testing and to automate the test generation process. The search issue is all about finding perfect results from a list of adversary results, which is handled by a fitness function to identify results. Software Testing is not just limited to the testing of an application or system but includes checking entirety of a system. Genetic Programming (GP) [1] is a type of Evolutionary Algorithm which is simulated by biological growth to search programs that perform certain user-defined tasks. This programming technique has been successfully applied to many fatigue problems present in software testing such as instinctive design, pattern recognition, and test suit generation [2].This suit of algorithms helps to automate the generation of basic test paths which includes several problems like data generation, sequence generation, test case derivation, and optimization. Recent studies stated a regeneration genetic algorithm which is proven to be operative and trivial for coverage-oriented software test suit generation. Issues related to software reusability can be resolved by the grouping of soft computing approaches like neural networks through software testing. Soft computing techniques such as GA are very much suitable for test size and coverage problems. Efforts are taken to develop the finest potential solutions for the automation in test suits and test sequence generation. Problems like test sequence, test data generation in white box testing and functional testing uses GA. In the current era of software development, test automation has a significant function in testing the software in its entirety. Test automation comes with its own challenges which include reusable scripts generation, recompiling the test scripts with modifications for different runs and rapid test development with least amount of development time and effort. Traditional methods such as randomized approaches, goal aligned techniques involve human intervention, development effort, cost. Limited Resources, missing the critical requirements and generation of redundant test cases are the prominent constraints in test case generation. To overcome the mentioned challenges test case generation methods needs enhanced algorithms.

II. Related Work

Test case generation techniques are classified into specification based which uses specification documents to derive the test cases, sketch based commonly work with diagrams such as UML, source code based where in test cases are derived using source code applicable to white box testing[3]. Study suggests test case generation to be a complex problem where in various strategies were proposed for the same. The algorithms GA, GA-NN and MA algorithms were applied in [4] which applies Machine Learning techniques to test generation process. Sketch based test case generation in combination with uml diagrams and state transition diagrams were proposed by [5].[6-8] Test case in combination with soft computing techniques such as Genetic Algorithm, Particle Swarm Optimization, Artificial Bee Colony derived a suitable results. The proposed approach considers test case generation process as a combinatorial optimization and the best feasible solution is in a set of discrete range. Combinatorial solutions were present in the literature, with different approaches: single objective optimization, multiple objective optimization. Test Case is a set of various combinations of input values which run on a scenario to produce the result and later decided accordingly. Hence the Test Case problem is (T, U, M, F), T is set of test instances can be considered as a test set, U is determinate set of solutions from the suite, given an instance x and a feasible solution y m is a measure on y.

Combinatorial approaches are a vital group of precise distinct enhancing approaches. These methods use successive analysis and exclusion of substitutes by Mikhalevich, Shor [9-11], the scheme is produces a group of additional schemes. For instance, dynamic encoding approaches, branch bound procedures can be defined in its framework. f is a fitness function usually considered to be a goal function which can be either min or max. Test case generation as a Single objective optimization [12] aims at achieving maximum fitness value such that the test suit derived will have the high probability of generating good code coverage. Consider T be the group of test cases $\{T_1, T_2, T_3 \ldots Tn\}$ and t is targets, the optimization problem is to generate a test suit to generate maximum fitness value.

Test case as a Multi objective optimization [13] here aims to maximize the fitness value considering along with code coverage the other parameters such as dependencies and all coverage criteria.

Max $F_i(X) = C_1(T_1, X), C_2(T_2, X), Cn(Tn, X)$, where

C_1, C_2, and Cn are the various coverage criteria.

Differential evolutionary techniques [14] uses multi-dimensional real valued function, generates new population based on existing solutions with a simple formulation.

III. Methodology

The current work proposes a System Graph, (graph representing the web application with annotations of number of link dependencies and number of data dependencies) for a program under test where the data and link dependencies of the web page are captured. As the test cases deals with the combination of input values the data dependencies [15] are considered to play a vital role. Further the link dependencies on the web page are prioritized. In combinatorial tests, Pairwise (T-way testing) contains on choosing a subsection of test cases that covers completely potential sets of arrangements, decreasing the amount of entire test cases to be executed, and growing the test efficiency by distributing its range, though using a minor pairs of test cases. This is mainly operative once the amount of variables surges, dropping extremely the amount of test cases. Pairwise testing [16, 17] which is proven to reduce the test cases considerable is been the next in the process so as to reduce the combination of input value pairs in the test suit generated. The proposed GA algorithm as specified in Fig. 1 generates and optimizes test cases specific to functional testing, is one of the randomized search procedures that have been established in a determination to emulate the method of regular selection and usual genetics. GA is proven to create first-class elucidations to optimization complications.

System Graph can be constructed either with the source code analysis [18, 19] and the web page itself. Since the data flows from one node to other node the resultant test cases after the Pairwise testing are considered, the same set of test cases are represented using a Graph for the logical representation of gene in our encoding phase of GA. Once the Graph generates the required paths as in Fig. 2, deliberated as test cases and genes are constructed.

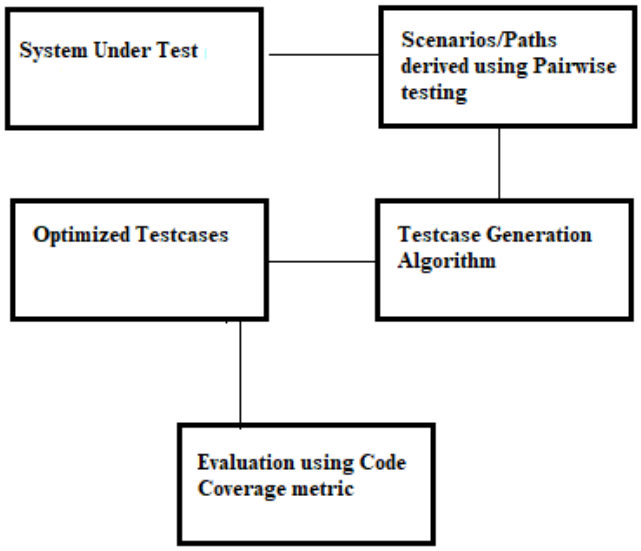

Fig. 1. Proposed System.

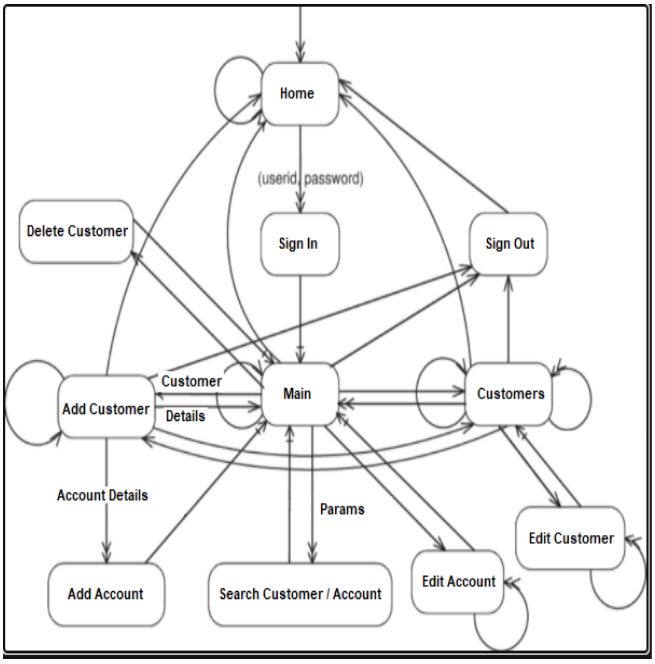

Fig. 2. Web Application Graph Model (System Graph).

A. Parameters Considered

Fitness value: Fitness is defined to give a value to each candidate solution which is considered to play a vital role in the search space. Fitness value [20] guides the whole Genetic Algorithm or a PSO algorithm in order to select the correct fit of individuals.

Code Coverage: Code Coverage [21] is considered to be a measure which of the test suite and the source code of the system covered with respect to this test suite.

Branch Coverage: Brach Coverage [22] is one more important method which ensures that the path/paths selected covers at least one branch, the branches true/false executed.

Dependency Coverage: Data dependencies and link dependencies [23] drive the extreme amount of test cases where page transits throughout the web application.

IV. Structure of Genetic Algorithm

Genetic Algorithm runs with an initial population of genes which are test cases. Fitness function is computed over the population to select set of chromosomes which will participate in the next generation population. Cross over and mutation operators are applied over the selected population to generate diversified range of population.GA stops once the population is either converged or for a specified number of iterations.

A. Genetic Algorithm

GA is a parameter coding technique which usually works on population of solutions and deterministic transitions. Considering the test case generation with respect to multi objective optimization PARETO [24] solutions and multi-criteria decision-aid technique is applied to select the finest solution. PROMETHEE technique [25] of decision is applied such that ranking amongst the individuals. Positive ranking is given as in Eq. 1, which expresses to what extent each alternative outranks all the others.

$$\sum_{b\in A, b\neq a}^{\pi(a,b)} \emptyset(a) = \frac{1}{n} - 1 \quad (1)$$

B. Genetic Algorithm Encoding

Each chromosome is encoded as a combination of pages and the data flow for each element to other element in a web page. We use a graph data structure to indicate the paths and web pages. The data flow from one element to other likely one page to other page is created. P1→P2→P3….Pn. From the Graph below sample genes considered:

geneA = {0, 1, 5, 8, 5, 9}
geneB = {5, 4, 7, 5, 6}
geneC = {1, 5, 2, 5, 6, 10, 6, 3}
geneD = {0, 1, 5, 3}

C. Fitness Function and Selection Mechanism

Tournament based selection [26, 27] is preferred over the roulette wheel selection as to lessen the risk of missing test cases. The primary fitness value is derived based on the valuation standard code coverage. If selected set of test cases covers the maximum code coverage are assigned to be highly probable. Secondary fitness value is dependent on the number of data dependencies and link dependencies of the given nodes. Individual gene with fitness f will succeed in the tournament of t individuals picked from the test suite with whole population given as in Eq. 2.

$$P(F)=MAX\ (F1,F2\ldots.FN)=X\ P(F<H)^{S-1}P(F) \quad (2)$$

where P(F) constitutes the probability. S denotes the genes having lower fitness score. The anticipated tournament succeed from a tournament size s is specified as in Eq. 3.

$$s\int f\ P(f\ s-1\ p(f))df \quad (3)$$

A test case is given a higher fitness value depending on the below functions.

Code Coverage of the chromosomes

nd→ Number of data dependencies,

nl→ Number of link dependencies,

tnd and tnl are the all-inclusive number of data dependencies and all-inclusive number of link dependencies contributed by the test case.

Test cases corresponding to the genes defined in the above section:

geneA :{ 0, 1, 5, 8, 5, 9}

TC1: P1→P2→P6→P9→P6→P10

geneB = {5, 4, 7, 5, 6}

TC2: P6→P5→P8→P6→P7

geneC= {1, 5, 2, 5, 6, 10, 6, 3}

TC3: P2→P6→P3→P6→P11→P7→P4

geneD = {0, 1, 5, 3}

TC4: P1→P2→P6→P4

Cross over: Single point crossover is considered initially to generate new population, if diversified range of population to be generated the other cross over operations can be applied.

TC1: P1→P2→P6→P9→P6→P10,

TC2: P6→P5→P8→P6→P7

TC3: P2→P6→P3→P6→P11→P7→P4,

TC4: P1→P2→P6→P4

TC11: P1→P2→P5→P8→P6→P7 (TC1&TC2)

TC12:P2→P6→P9→P6→P10 (TC1&TC3)

The mutation process [28] is to maximize the chance of complete search space in the algorithm, a predefined mutation probability [29-30] is calculated for each chromosome, and score is arbitrarily engendered to relate the mutation probability to resolve for the mutation process. Sample of the test cases after the crossover operation and mutation operation.

From the above generated test cases:

TC21: P4→P6→P9→P6→P10 (TC1 and TC4)

Acceptance: As the mutation and crossover involve certain level of uncertainty, the off springs may or may not be superior to parent chromosomes. Hence fitness needs to be calculated for acceptance.

Stop criteria: for a specified number of maximum generations the GA is executed, based on the fitness and code coverage the GA is stopped.

ALGORITHM - TEST CASE GENERATION

Input:
Program under test
Initial set of paths (Test Cases) from the System Graph for Web application.
Initial set of paths (Test Cases) from the Program Graph for console programs.
Output:
Set of optimized paths (Test Cases),
{P1, P2, P3…Pn}, Code Coverage, Fitness value.

Initialization phase:
Build a DLDG graph for the corresponding program under test.
Generate initial population of genes
{TC1, TC2, TC3…TCn},
Apply Pairwise testing to generate genes
{TCm1, TCm2, TCm3…TCmn}

GA Algorithm:
gen1=1, max_gen
Current_population:
{TCm1, TCm2, TCm3…TCmn}=
{P1→P2→P3..Pi}, (initial set of paths)
While (gen1≤ max_gen)
Begin
for each gene $gene_i$ in Current_ population
{$gene_1$, $gene_2$,$gene_3$…$gene_n$}
Calculate the fitness_ value Fi as specified in Eq.4

$Fi = \sum_{i=0}^{n} Ci + TCi((nd + nl)/(tnd + tnl))$ Eq. (4)

Ci is the code coverage of the test suit, nd, nl,tnd,tnl as stated in the fitness and selection mechanism.
for each gene{$gene_i$ }
If (fitness_value is in the range)
Select the gene {$gene_i$} based on Tournament based selection
Apply crossover operation to generate the new genes
Apply mutation operation to change the gene
Add the above population to the current_ population
End

V. Experiments and Evaluation

Experiment 1:

Triangle classification problem where in the input is considered for three sides of a triangle and the output details the type of a triangle. SideA, SideB, SideC for the first generation was chosen randomly as specified in Table I, these values were further selected to be part of parent chromosomes and underwent GA operations using the fitness function and pairwise testing described in the above algorithm. Pairwise testing values were obtained using online Pairwise online tool. Code coverage from the second generation was noted and specified in Table II. NUnit coverage tool is used to record the code coverage of the test suit. The tables provide the data obtained as a result of our methodology in Fig. 1.

Fig. 3 illustrates the tests vs coverage in terms of line and branch coverage for the values specified on the horizontal axis.

TABLE I. Generation- 1

SideA	*SideB*	*SideC*
0	2	1
2	5	1
1	1	5
5	2	1

TABLE II. Generation-2

Test case SideA,SideB,SideC	*Branch coverage*	*Line coverage*
1,2,1	33.33%	40.90%
1,0,0	33.33%	31.81%
1,5,0	25%	31.81%
1,1,2	33.33%	31.81%
1,1,0	25%	31.81%
1,0,2	33.33%	31.81%
2,1,1	41.66%	40.90%
2,5,2	41.66%	40.90%
2,0,1	33.33%	31.81%
2,1,1	33.33%	31.81%
2,2,0	25%	31.81%
5,5,1	41.66%	40.90%
5,0,1	33.33%	31.81%
5,1,1	33.33%	31.81%
5,1,0	25%	31.81%
5,2,1	50%	45.45%
5,1,2	50%	45.45%
0,0,1	25%	31.81%
0,1,0	25%	31.81%
0,1,1	25%	31.81%
0,2,2	25%	31.81%
0,5,1	25%	31.81%

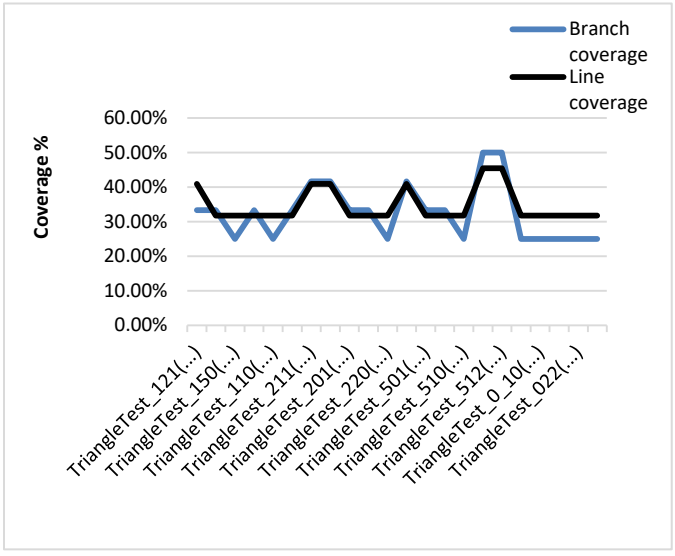

Fig. 3. Tests vs. Coverage.

The sample values after processing and normalized values achieved the below result as shown in Fig. 4.

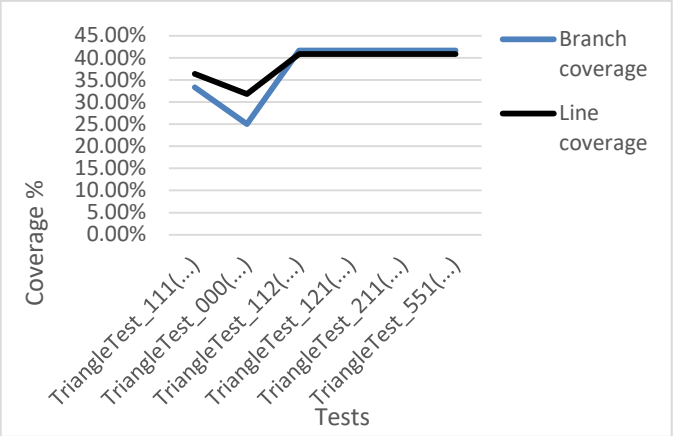

Fig. 4. Tests vs. Coverage.

The results after eight generations achieved a consistent result which achieved 88.20% of code coverage and are depicted in the Fig. 5.

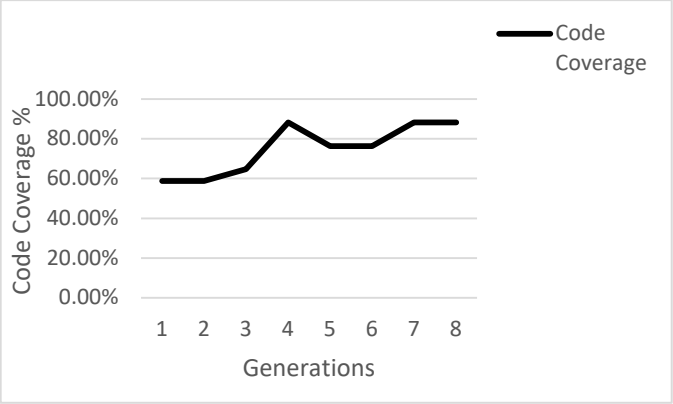

Fig. 5. Generation vs. Code Coverage.

Experiment 2:

The source code for a simple web application was considered for experimental evaluation and random test cases were generated. This was a Web based application as represented in Fig. 2, the automated test cases were captured using selenium IDE. Selenium IDE is basically a record and playback tool, the test cases generated by Selenium IDE are saved and deployed as JUnit, NUnit test cases. The sample test cases were run through NUnit code coverage [31, 32] which achieved the following result over the main modules like performing an insertion and deletion of the records of customers.

Proposed GA Algorithm was then executed on the same set considering few sample test cases from the above document, which achieved the following result. At each iteration the fitness value is generated using the fitness function and the genes are allotted the ranking as per selection criteria discussed previously. The set of genes which are valid and invalid is checked manually which can be automated further. Hence the genes undergo a preprocessing phase for the mentioned. Considering the above mentioned geneA, geneB, geneC, geneD, Sample of genes generated by GA algorithm for three of the generations are as mentioned below.

Test cases derived for a sample of three generations

Generation 0: Random population considered from the Fig. 3 are:

[0, 1, 5, 8, 5, 9]
[5, 4, 7, 5, 6]
[1, 5, 2, 5, 6, 10, 6, 3]
[0, 1, 5, 3].

After processing with the selection, crossover and mutation operation the following were the chromosomes generated for second generation.

Generation 1:

[0, 1, 5, 5, 6, 9]
[5, 4, 7, 8, 5]
[0, 1, 5, 8, 5, 9]
[5, 4, 7, 5, 6]
[0, 1, 5, 5, 6, 9]
[5, 4, 7, 8, 5]
[0, 1, 5, 8, 5, 9]
[5, 4, 7, 5, 6]
[0, 1, 5, 3, 6, 10, 6, 3]
[1, 5, 2, 5]
[0, 5, 2, 5, 6, 10, 6, 3]
[1, 1, 5, 3]
[0, 5, 5, 3, 6, 10, 6, 3]
[1, 1, 2, 5]
[0, 5, 5, 5, 6, 10, 6, 3]

Generation 2:

[0, 5, 5, 5, 6, 10]
[0, 1, 5, 8, 5, 9, 6, 3]
[0, 1, 5, 8, 5, 9]
[0, 5, 5, 5, 6, 10, 6, 3]
[0, 1, 5, 5, 6, 10]
[0, 5, 5, 8, 5, 9, 6, 3]
[0, 1, 5, 8, 5, 9]
[0, 5, 5, 5, 6, 10, 6, 3]
[1, 1, 2, 3, 6]
[5, 4, 7, 5]
[1, 4, 7, 5, 6]
[5, 1, 2, 3]
[1, 4, 2, 3, 6]
[5, 1, 7, 5]
[1, 4, 2, 5, 6]
[5, 1, 7, 3]

The above values after preprocessing where in repeated genes and invalid genes were processed and further reduced. Validity and invalidity of the genes were verified based on the data associated with the genes, for instance the path from P1→P2 is valid based on data which were minimized using pairwise testing. Considering P1 to be a Login page the page transits to other page if P1 {data} is valid, if P1 {data} is not

valid the page transits to other page. Fig. 6 and Fig. 7 depicts the graph Tests vs. coverage, the very first initialization of the test suit is chosen randomly specified with values in the horizontal axis, where the line coverage and branch coverage are proportional, the intermediate tests didn't achieve the coverage but stabilized in due evolution with Genetic Algorithm.

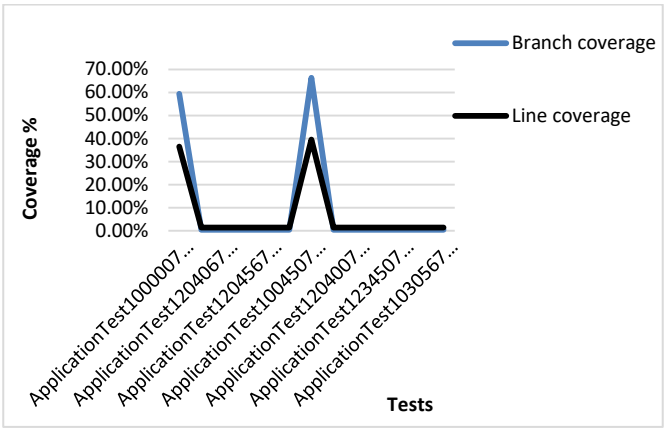

Fig. 6. Tests vs. Coverage for First Generation (Random Test Cases).

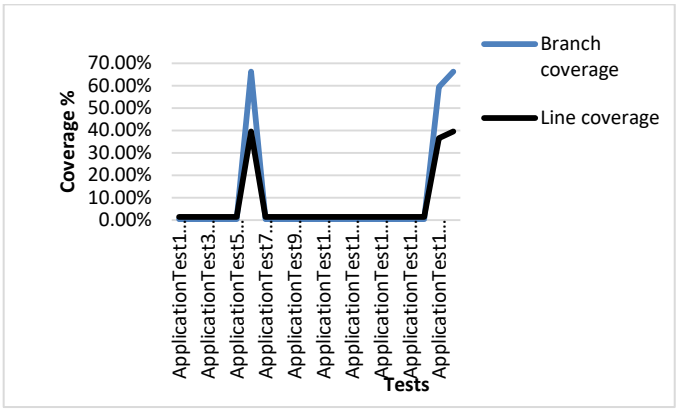

Fig. 7. Tests vs. Coverage for Generation 2.

VI. Future Scope

The future work of the proposed work is to evaluate with large scale web applications and console programs. Though the current approach proposes an automated solution, the pairwise integration, validation for each run with respect to the test cases is done manually. The work can be extended with a complete automated integrated solution for generating test cases for the entire process.

VII. Conclusion

This paper proposes an automated solution for Test case generation problem by means of Integrated Pairwise Genetic algorithm. A set of optimized test cases after Pairwise testing are considered as initial population for the GA. Considerably less genes were initiated which leads gradually to huge amount of test suites. Code coverage was measured at every generation and based on fitness values the parent genes were selected and then were involved in the generation process. When the evaluation metric code coverage is compared with random generation of test cases and GA, the results show that GA has considerably increased the fitness value and code coverage. Further our work requires and automated integrated solution for the whole process.

References

[1] Katoch, S., Chauhan, S.S. & Kumar V, "A review on genetic algorithm: past, present, and future," Multimed Tools Appl (2020). https://doi.org/10.1007/s11042-020-10139-6.

[2] TianTian and Dunwei Gong. 2016., "Test data generation for path coverage of message-passing parallel programs based on co-evolutionary genetic algorithms," Automated Software Engg. 23, 3 (September 2016), 469–500. DOI:https://doi.org/10.1007/s10515-014-0173-z.

[3] NichaKosindrdecha and JirapunDaengdej, 2010, "A Test Case Generation Process and Technique," *Journal of Software Engineering*, 4: 265-287.

[4] M.R. Keyvanpour, H. Homayouni and HaseinShirazee, 2011, "Automatic Software Test Case Generation," Journal of Software Engineering, 5: 91-101.DOI**:** 10.3923/jse.2011.91.101

[5] Khurana N, Chillar RS, " Test Case Generation and Optimization using UML Models and Genetic Algorithm," Procedia Computer Science [Internet]. 2015;57:996–1004.Available from: .http://dx.doi.org/10.1016/j.procs.2015.07.502.

[6] Rijwan Khan, Mohd. Amjad, "Automatic test case generation for unit software testing using genetic algorithm and mutation analysis," 2015*IEEE UP Section* Conference on Electrical Computer and Electronics (UPCON).

[7] Shveta Parnami, KrishnaSwaroop Sharma, "Empirical Validation of Test Case Generation based on All- Edge Coverage Criteria,"*International Journal of Computer Applications*,September 2015.

[8] Baswaraju Swathi, Dr.Harshvardhan Tiwari, "Genetic Algorithm Approach to Optimize Test Cases," International Journal of Engineering Trends and Technology 68.10(2020):112-116.

[9] Hulme A, Mclean S, Salmon PM, Thompson J, Lane BR, Nielsen RO, " Computational methods to model complex systems in sports injury research: agent-based modelling (ABM) and systems dynamics (SD) modelling," Br J Sports Med. 2019 Dec;53(24):1507-1510. doi: 10.1136/bjsports-2018-100098. Epub 2018 Nov 17. PMID: 30448782.

[10] Abu Sharkh, M., Shami, A. &Ouda, A, "Optimal and suboptimal resource allocation techniques in cloud computing data centers," J Cloud Comp 6, 6 (2017). https://doi.org/10.1186/s13677-017-0075-2.

[11] Islam, M.R., Mahmud, M.R. &Pritom, R.M, "Transportation scheduling optimization by a collaborative strategy in supply chain management with TPL using chemical reaction optimization," Neural Comput&Applic 32, 3649–3674 (2020). https://doi.org/10.1007/s00521-019-04218-5.

[12] Ram Krishna Rathore, Kaushal Sharma , Amit Sarda, "An Adaptive Approach for Single Objective Optimization,"Ram Krishna Rathore et al Int. Journal of Engineering Research and Applications ,ISSN : 2248-9622, Vol. 4, Issue 2(Version 1), February 2014, pp.737-746.

[13] AnnibalePanichella , Fitsum MesheshaKifetew , "Automated Test Case Generation as a Many-Objective Optimization Problem with Dynamic Selection of the Targets,"IEEE Transactions on Software Engineering (Volume: 44 , Issue: 2 , Feb. 1 2018).

[14] Libiao Zhang, Xiangli Xu, Chunguang Zhou, Ming Ma, Zhezhou Yu, "An Improved Differential Evolution Algorithm for Optimization Problems,"Advances in Computer Science, Intelligent System and Environment.

[15] HasanUral,KassemSaleh, Alan W Williams, "Test generation based on control and data dependencies within system specifications in SD," Computer Communications 23(7):609-627.

[16] Kamal Z Zamli, "T-Way Strategies and Its Applications for Combinatorial Testing," International Journal on New Computer Architectures and Their Applications (IJNCAA)1(2): 459-473The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085).

[17] Feng Duan et al, "An Approach to T-Way Test Sequence Generation with Constraints," 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).

[18] Haralambi Haralambievet al, "Applying source code analysis techniques: A case study for a large mission-critical software system," 2011 IEEE EUROCON - International Conference on Computer as a Tool.

[19] Rahma Mahmood, Qusay H. Mahmoud, "Evaluation of Static Analysis Tools for Finding Vulnerabilities in Java and C/C++ Source Code," arXiv.org cs, 1805.09040,Cornell University.

[20] Rahm Mitrabinda Ray, "PSO based test case generation for critical path using improved combined fitness function," Journal ,Volume 32, Issue 4, May 2020, Pages 479-490.

[21] Marko Ivanković et al, "Code Coverage at Google," Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, pp. 955-963.

[22] Giovanni Grano et AL,**"**Branch coverage prediction in automated testing,**"***Journal of Software Evolution and Process*, 08 March 2019.

[23] MatteoBiagiolaetal, "Web Test Dependency Detection," *Coronell University,arXiv:1905.00357*.

[24] Mark Harman, KiranLakhotia, Phil McMinn, "A Multi–Objective Approach To Search–Based Test Data Generation," GECCO'07, July 7–11, 2007.

[25] Sun Zhaoxu , Han Min et al, " Multi-criteria Decision Making Based on PROMETHEE Method," 2010 International Conference on Computing, Control and Industrial Engineering.

[26] YongshengFang,Jun li , "A Review of Tournament Selection in Genetic Programming,". Cai et al. (Eds.): ISICA 2010, LNCS 6382, pp. 181–192, 2010.c©Springer-Verlag Berlin Heidelberg 2010.

[27] ArtemSokolov,DarrellWhitley, "Unbiased tournament selection," Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005.

[28] R. Tinos,A.C.P.L.F. de Carvalho, "A genetic algorithm with gene dependent mutation probability for non-stationary optimization problems," IEEE, Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753).

[29] Jürgen Hesser, Reinhard Männer, "Towards an optimal mutation probability for genetic algorithms,"Genetic Algorithms Genetic Algorithm Theory, International Conference on Parallel Problem Solving from Nature,Springer,PPSN 1990: Parallel Problem Solving from Nature pp 23-32.

[30] BaswarajuSwathi, Harshvardhan Tiwari, "Test Case Generation Process using Soft Computing Techniques," International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278- 3075, Volume-9 Issue-1, November 2019.

[31] Boyuan Chen et al,"An Automated Approach to Estimating Code Coverage Measures via Execution Logs," 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE).

[32] Matina C. Donaldson-Matasci, Carl T. Bergstrom, and Michael Lachmann,"The fitness value of information," *Oikos,PMC* 2015 Apr3.