IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-6, June 2018 E-ISSN: 2347-2693

An Algorithm for Test Case Reduction in Regression Testing

Arzu[1*], S. Singh[2]

[1] Dept. of Computer Science and Engineering, DCRUST, Murthal, Sonepat, India
[2] Dept. of Computer Science and Engineering, DCRUST, Murthal, Sonepat, India

[*]Corresponding Author: aarzooparashar4752@gmail.com

Available online at: www.ijcseonline.org

Accepted: 11/Jun/2018, Published: 30/Jun/2018

***Abstract*—** Testing is one of the most critical and time consuming phase in development of a software and when it comes to regression testing in which compatibility is checked of the previous code with the updated one. It definitely increases the size of test case and budget so, to decrease the number of test cases in regression testing this paper presents a QBGA (Queen Bee Genetic Algorithm) technique for test case reduction and also increases the coveragence that would makes a software more efficient. When it is in contrast with the existing GA algorithm, the number of test cases is found to be reduced and covered area is enhanced and results are found to be better.

***Keywords*—**RegressionTesting,TestCaseReduction,Coveragence

I. INTRODUCTION

Importance of Software Testing

Software testing is very important quality assertiveness scheme which make sure that a quality software system is delivered by examine a program to recognize errors and to guarantee the correctness and reliability of the software product. Due to the increasing demand of software systems it continuously evolves several changes like system functionality, technologies and current demands which affect the existing modules of a system under test [1], [2].

Types of software testing:

Static and Dynamic testing

Design documentation, software requirement specification (SRS), code documentation examine through work documents in static software testing. Testing is done through walkthrough processes, inspections, informal and formal reviews and static code reviews. Dynamic testing is done by executing the code averse to a given set of inputs and conditions. System testing, integration testing and unit testing includes in dynamic testing techniques.

Functional and Non Functional Testing

This is a kind of testing which is done when a tester does not know about the initial functioning of the given program and which algorithms are involved in it. In white box testing in which the testers know exactly about the structuring of the program to be tested and it is also called transparent testing because the tester is aware of the inner functioning of the program [3].

Regression Testing

Regression testing reruns some of the already executed test cases with the aim of checking whether previously fixed faults are remerged and to ensure that the new changes do not have a refusing on the existing system software [4].

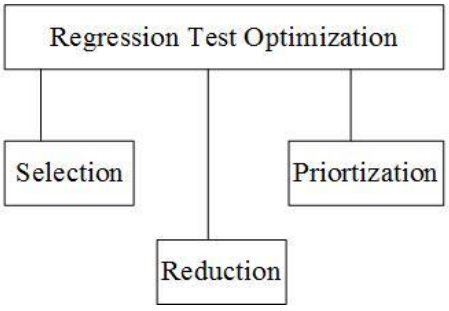

Figure 1. Regression Testing Optimization

1. Selection Technique

It chooses a test suite subset that is used during the development process and wields that subset to test the new modification in the program. In this method it divides the test suite in to three parts:

(a) Reuse the test cases
(b) Re test the test cases
(c) Absolute test cases

These methods helps to pick a subset of test cases without picking all the test cases only a subset is selected to test the modified program which diminishes time and efforts but somewhere it decreases the test effectiveness [6].

2. Prioritization Technique

In this technique testers prioritizes the test cases in regression testing in reference to several factors such as cost benefit, execution time , total efforts, etc. that enhances a fault detection rate of a test suite [7]. This technique uses least recently used method to prioritize the test cases. There are two types of prioritization:

(a) General Prioritization

(b) Version Specific Prioritization

In General prioritization it selects test cases in such a order that it has highly effective and version specific software is in reference with a particular version of software [8].

3. Reduction Technique

In test suite reduction technique removes the irrelevant test cases and its major aim is to enhance the capability and compatibility of the existing test cases with the modified program. There are three types of methods which are:

(a) Retest all test cases [R1]

In reset all test cases method that validates and verifies the software moderation by testing each of the test case in a particular test pool for every program. It records the test suite size increases which means selected test cases can also grows. When size is to big it becomes very difficult to fix that problem. There are several types of approaches proposed for selection method like random technique, regression test model, minimization and prioritization techniques, etc. R1 works very well when dealing with small set of test suites and whenever it deals with large number of test suites T' it drops [5].

(b) Random deletion [R2]

Random detection is used when the size of test suites are really large. It fixes the problems by using randomly detaching technique in which numbers of test cases are reduced without avoiding the errors. This technique helps the software developers to save time in testing phase of the software development and its performance is also better as compare to R1. This method reduces the number of chosen test cases with less number of errors.

(c) Test Reduction in Regression Testing [R3]

This technique is used for choosing a set of test suites to keep away programming errors or bugs when size is in control. It solves the crucial problems in the maintenance phase and it decreases the size of T' by picking selected test suites. This technique gives better results when compare to R1 and R2 methods [9].

II. RELATED WORK

Aiming on the importance of software testing during the software development process and addressing the issues of testing regarding cost and time [2] experiences that testing can enhance any software performance and evaluates its functionality several types of testing is done to test a software such as black box testing in which a tester verifies against a suite of test cases that would already know the right outputs [3] Regression testing does not compromises the existing software functionality and always reruns the programs whenever there is an update to check their compatibility with each other. It uses test suite prioritization by application navigation tree mining through which a tree is constructed and it choose only optimal solutions returned by CI approach and fuzzy logic [20,21] another reduction algorithm in which concept of greedy technique and AIA are combined together which approaches the calculation, coverage and enhance the quality of testing in less time. It also approaches to the fault localization regression test cases reduction techniques [11] Neha Sharma and Sujata [7] presented a technique GA for test case prioritization by implementing stochastic optimization when deal with fitness value and it is evaluated that metaheuristic genetic algorithm improves the efficiency as compared to the existing algorithms in much better way. Indumathi and S.Madhumathi [15] has proposed MFTS (Maximum frequent test set) algorithm in which number of test cases decreases and increases the fault detection rate. This method eliminates the redundant test cases and set their priorities and then executes them.

III. METHODOLOGY

In QBGA algorithm the very first one step is to initializing population so let us take a s x t requirement matrix in which binary values are present that is 0 and 1's. Suppose there is 1 present that means the test case is selected and whenever there is a 0 that means the test case would not be selected. The test sets are represented in row as T= {T1, T2, T3, T4…..Tx} and number of requirements satisfied represents in columns as R= {R1, R2, R3, R4….RX}. There are 10 numbers of requirements satisfied and 10 number of test cases in given table.

Queen Bee Genetic Algorithm based on the concept of natural selection of the fittest individual and it re-establishes the theory of evolution. QBGA uses biological processes and assumes that a problem is resolved by a potential where individual are represented as a set of parameters structured in the form of binary string values. Its main aim is searching and ascertain out optimal solution using fitness value before choice is making. This algorithm uses the operators such as selection, crossover and mutation and every evolution consists of population from which individuals are selected arbitrary [5, 10, 11].

Table 1. A set of dummy values in binary form

Test Cases	Number of Requirements Satisfied									
	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
T_1	0	0	0	0	0	1	0	0	1	0
T_2	0	0	0	0	0	0	1	0	0	1
T_3	1	0	1	0	0	1	0	0	1	0
T_4	0	1	0	1	0	1	0	1	0	0
T_5	0	0	0	0	0	0	0	1	0	1
T_6	0	0	1	0	1	0	0	0	1	0
T_7	1	0	1	0	1	0	0	0	0	0
T_8	0	0	0	0	0	0	0	0	0	1
T_9	0	0	0	0	0	0	1	0	0	0
T_{10}	0	0	0	0	0	0	0	0	1	0

A. Population Initialization

Every individual of initial population presents a test suite. Test suite pond is used to select initial population randomly and each suite contains a test case set. [13].

T_7	T_{13}	T_5	T_{11}	T_{16}	T_9	T_4	T_{15}	T_3	T_8

Figure 2. Initialization of Population

B. Fitness Value

Fitness value of each test case or individual is computed while performing an operation amid all requirement suites of individual test case and after that the result of the fitness value is converted in to the form of percentage

$F(x_1)$ = (No. Of requirements fulfil / Total no. Of requirements) × 100

$F(x_2) = W_1 \times obj_1 + W_2 \times obj_2$

W_1 and W_2 are normalised constants

Obj_1 = Percentage of selected test cases

Obj_2 = Percentage of fault coverage

C. Selection

Selecting technique is used to choose all possible solution to produce the solution of coming generation and the best selected once must create new off springs or child's. In QBGA one parent is selected as Queen which is the best one and second parent is selected as random one. [14].

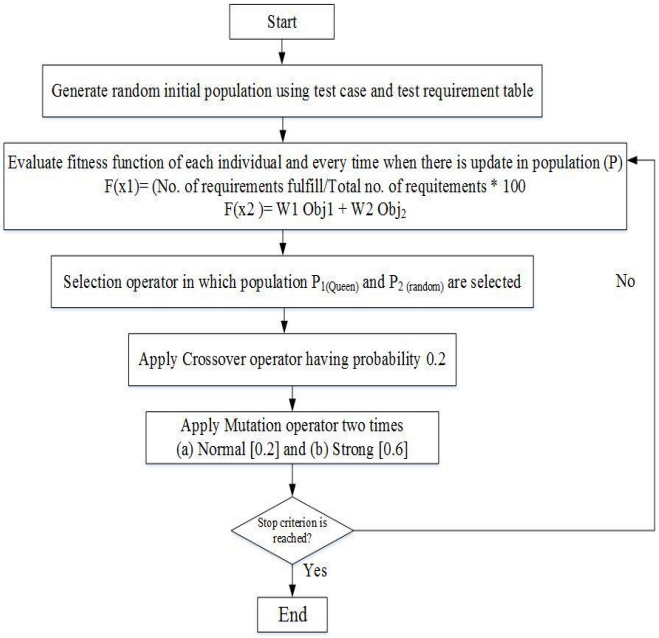

Figure 3. Flow Diagram of QBGA Algorithm

D. Crossover Operator

After the selection of test sets we applied crossover with crossover probability of to generate new offspring from the selected parents. In QBGA one parent who is male one is selected randomly and another parent which is female is selected as the best one also called as Queen. $P_1 \times$ Queen (best) →C

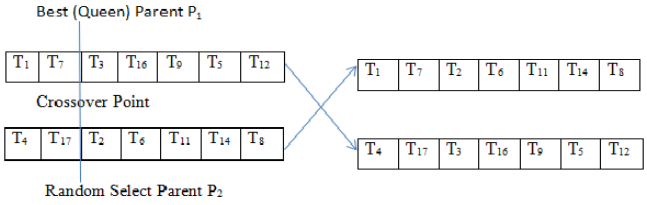

Figure 4. Crossover between P_1 and P_2

For example: Let's take Parent 1 and Parent 2 two individuals represented as

$Parent_1 = < T_1, T_7, T_3, T_{16}, T_9, T_5, T_{12} >$ and $P_2 = < T_4, T_{17}, T_2, T_6, T_{11}, T_{14}, T_8 >$

$Parent_1$ and $Parent_2$ could be crossed according to the crossover point and then produce two Childs which results into P_1 becomes, $P_1 = < T_1, T_7, T_2, T_6, T_{11}, T_{14}, T_8 >$ and P_2 becomes

$Parent_2 = < T_4, T_{17}, T_3, T_{16}, T_9, T_5, T_{12} >$

E. Mutation

Mutation includes random modifications and it is operate to grasp all the redundant test cases which is available in test

pool. In the mutation process values are swapped according to the probabilities as one set of test suite having high mutation or normal mutation.

There are two types of mutation performed in this algorithm
(a) Normal (approx. = 0.2)

(b) Strong (approx. = 0.6)

In normal mutation having lower probability like 0.1, 0.2, 0.3 .. and a swap operator is used to swap these values and in strong mutation probabilities such as 0.5, 0.6, 0.7 …that means a large number of test cases or values are swapping.

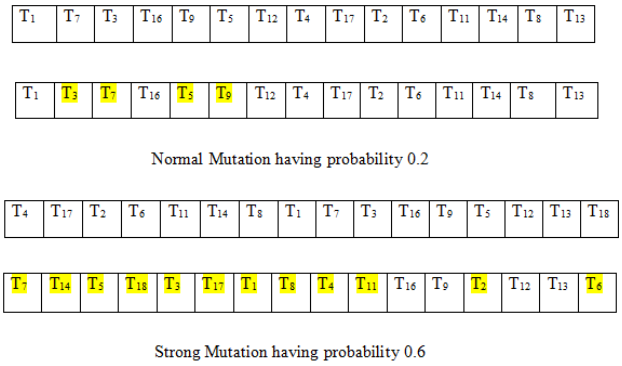

Figure 5. Normal and Strong Mutation Operation

Algorithm (QBGA)
Inputs:
P : Size of Population
T : Set of test cases
Cp : Crossover Probability
Mp : Probability in mutation operation (normal, strong)
G : Number of Generations
Output:
$P_i \leftarrow$ population gen (t,tc,tsp)
For i = 1 to G
$F_i \leftarrow$ fitness eval [(Fx_1) and $(Fx_{2)}$]
Fx_1 = (No. Of req fulfill/ Total No. Of req) × 100
$Fx_2 = (W_1 \times Obj_1 + W_2 \times Obj_2)$
W_1 and $W_2 \leftarrow$ Normalised Constants
$Obj_1 \leftarrow$ Per of selected test cases $[S_i]$
$Obj_2 \leftarrow$ Per of fault coverage $[F_i]$
$P_{i+1} \leftarrow$ add one random (r_1) and
one queen (q_1) [$P_{1 \times queen} = G$]
$P_1 \leftarrow$ queen(q_1) select parent $[p_i]$
$P_2 \leftarrow$ rand select parent $[p_i]$
CM $\leftarrow$ crossover ($P_1 \times P_2 = C_p$)
$M_1 \leftarrow$ Mutation (C_1, M_P)
$M_2 \leftarrow$ Mutation (C_2, M_P)
Normal Mutation $\leftarrow$ Prob [0.2]
Strong Mutation $\leftarrow$Prob [0.6]
$G \leftarrow G+1$
$F_{G+1} \leftarrow$ eva fitness (G+1,t,fl,fsl)
$T_G \leftarrow$ select best child $(F_{G+1,G+1})$
Return T_G [15]

IV. RESULTS AND DISCUSSION

It should include important findings discussed briefly. Wherever necessary, elaborate on the tables and figures without repeating their contents. Interpret the findings in view of the results obtained in this and in past studies on this topic. State the conclusions in a few sentences at the end of the paper. However, valid colored photographs can also be published Algorithm proposed in this work is used to reduce the number of selected test cases and enhance the coverage of test cases. It is implemented using Queen Bee Genetic Algorithm which improves the quality of initial population by selecting one parent as 'Queen' and evaluates fitness function whenever population is updated. There are two types of mutation operator performed in this algorithm one is normal and another one is strong mutation which helps in removing the redundant test cases and improves the quality of testing in limited timed time and budget. This algorithm is applied to various data sets of binary values of test cases like a data set 'flex v1', 'printtokens' and schedule etc. and it is observed that in flex v1 there is 567×19 matrix on which GA and QBGA both were applied to check the performance of both algorithms and it is observed that QBGA performs in a much better way as it reduced the test cases by 185 whether GA reduces the test cases and selects 226 test cases than it is observed that in GA numbers of selected test If a matrix of 10×10 considered as set of dummy values in which number of selected test cases and coverage required is taken cases are 5 while QBGA selects only 3 test cases and in GA required coverage is 80% and QBGA covers 90% of the test sets.

Table 2. Results of Dummy Data Set Matrix

Dummy Data Set	10x10	
Approach	**Num Selected Test cases**	**Req Coverage%**
GA-BASE	5	80
QBGA-Proposed	3	90

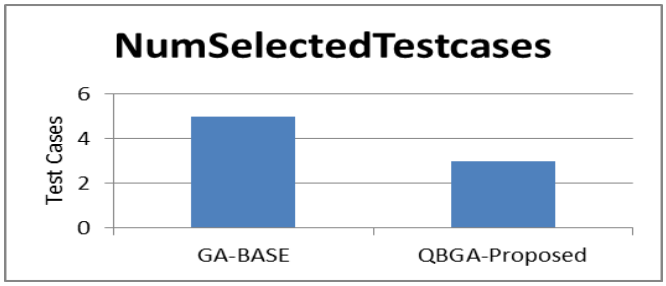

Figure 9. Reduction Analysis Graph

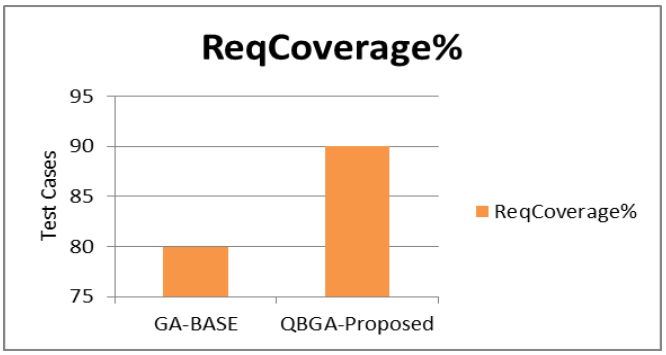

Figure 10. Coverage% Analysis Graph

Table 3. Results of Printtokens (200×200) Matrix

Printtokens	200x200	
Approach	**NumSelectedTestcases**	**ReqCoverage%**
GA-BASE	82	100
QBGA-Proposed	73	100

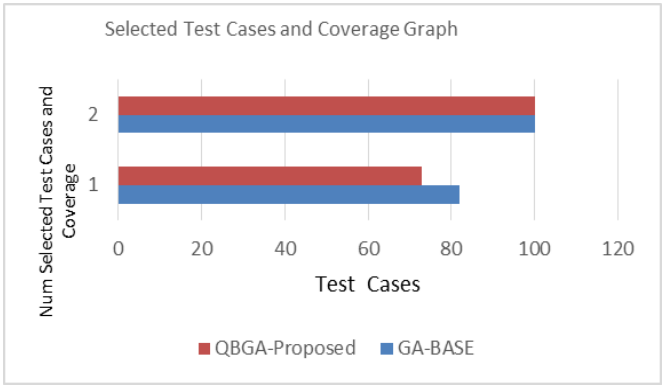

Figure 11. Performance Analysis Graph

Table 4. Comparative Output Table of GA and QBGA

Approah		Num Selected Test Cases					Req Coverage %			
GA Based		5	226	87	82		80	84.12	99.5	100
QBGA Proposed		3	185	64	73		90	84.21	99.5	100

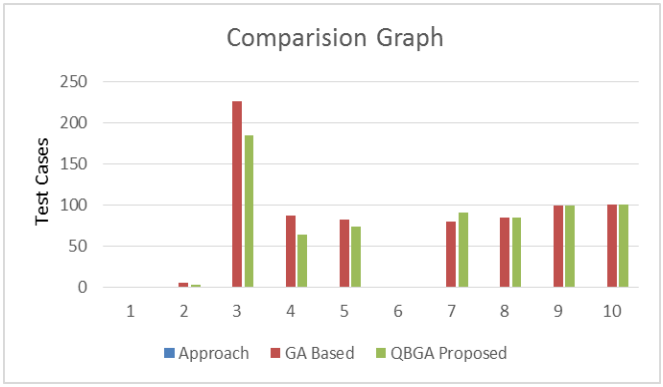

Figure 12. Comparison between GA and QBGA

Comparative analysis of various matrices together in which main focus is on number of selected test cases and coverage of entire test suites on the basis of this a graph is plotted. These graphs shows that queen bee genetic algorithm select less number of test cases as compare to genetic algorithm and cover more number of test suites that is coverage.

V. Conclusion and Future Scope

This paper presents an algorithm for test case reduction in regression testing and its implementation. After implementation the proposed algorithm is compared with the Genetic Algorithm. QBGA algorithm is used for test case reduction in regression testing. The test cases are minimized greatly which reduces the cost of testing in the development phase. This technique of reduction provides an effective number of test case reductionS in test suites and also increases the coverage area which improves the efficiency and quality of the software. On the basis of defect detection capability many interesting result have been collected but the test case reduction in regression testing using queen bee genetic algorithm has not been traversed. On the basis of defect detection capability many interesting result have been collected but the test case reduction in regression testing using queen bee genetic algorithm has not been traversed. So there is still space for the researchers to demonstrate and validate the queen bee genetic algorithm based approach to reduce the set of test cases on the premise of statement coveragibility.

References

[1] Khan, S. U. R., Lee, S. P., Javaid, N., & Abdul, W. (2018). "A Systematic Review on Test Suite Reduction": Approaches, Experiment's Quality Evaluation, and Guidelines. *IEEE Access*, *6*, 11816-11841.

[2] de Souza, D. M., Oliveira, B. H., Maldonado, J. C., Souza, S. R., & Barbosa, E. F. (2014, October). "Towards the use of an automatic assessment system in the teaching of software testing." In *Frontiers in Education Conference (FIE), 2014 IEEE*(pp. 1-8). IEEE.

[3] Velmurugan, P., & Mahapatra, R. P. (2016, July). "Effective testcase retaining using branch coverage technique." In *Applied and Theoretical Computing and Communication Technology (iCATccT), 2016 2nd International Conference on*(pp. 504-508). IEEE.

[4] Lawanna, A. (2017, October). "Extended regression test model for test suite reduction." In *Consumer Electronics (GCCE), 2017 IEEE 6th Global Conference on* (pp. 1-5). IEEE.

[5] Jyoti, K. S. (2014). A Comparative Study of Five Regression Testing Techniques: A Survey. *International Journal of Scientific & Technology Research*, *3*(8).

[6] Yamu, A., Cingiz, M. ., Biricik, G., & Kalpsz, O. (2017, June). "Solving test suite reduction problem using greedy and genetic algorithms." In *Electronics, Computers and Artificial Intelligence (ECAI), 2017 9th International Conference on* (pp. 1-5). IEEE.

[7] Sharma, N., & Purohit, G. N. (2014, December). "Test case prioritization techniques "an empirical study". In *High Performance Computing and Applications (ICHPCA), 2014 International Conference on* (pp. 1-6). IEEE.

[8] Ansari, A. S., Devadkar, K. K., & Gharpure, P. (2013, December). "Optimization of test suite-test case in regression test."In *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on* (pp. 1-4). IEEE

[9] Lawanna, A. (2017, October). "Extended regression test model for test suite reduction." In *Consumer Electronics (GCCE), 2017 IEEE 6th Global Conference on* (pp. 1-5). IEEE.

[10] Nagar, R., Kumar, A., Kumar, S., & Baghel, A. S. (2014, September)."Implementing test case selection and reduction techniques using meta-heuristics."In *Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference-* (pp. 837-842). IEEE.

[11] Hui, Z. (2016, June). "Fault Localization Method Generated by Regression Test Cases on the Basis of Genetic Immune Algorithm."In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual* (Vol. 2, pp. 46-51). IEEE.

[12] You, L., & Lu, Y. (2012, May)."A genetic algorithm for the time-aware regression testing reduction problem." In *Natural Computation (ICNC), 2012 Eighth International Conference on*(pp. 596-599). IEEE.

[13] Hui, Z. (2016, June). "Fault Localization Method Generated by Regression Test Cases on the Basis of Genetic Immune Algorithm."In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual* (Vol. 2, pp. 46-51). IEEE.

[14] Mohapatra, S. K., & Prasad, S. (2014, March). "Minimizing test cases to reduce the cost of regression testing." In *Computing for Sustainable Global Development (INDIACom), 2014 International Conference on* (pp. 505-509). IEEE.

[15] Indumathi, C. P., & Madhumathi, S. (2017, May)."Cost aware test suite reduction algorithm for regression testing."In *Trends in Electronics and Informatics (ICEI), 2017 International Conference on* (pp. 869-874). IEEE.

[16] Hui, Z. (2016, June). "Fault Localization Method Generated by Regression Test Cases on the Basis of Genetic Immune Algorithm."In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual* (Vol. 2, pp. 46-51). IEEE.

[17] Rosero, R. H., Gómez, O. S., & Rodríguez, G. (2017). "Regression Testing of Database Applications Under an Incremental Software Development Setting."*IEEE Access*, *5*, 18419-18428

[18] Liu, P. (2014, June). "An efficient reduction approach to test suite. In *Software Engineering,"Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on* (pp. 1-5). IEEE.

[19] Alagöz, I., Herpel, T., & German, R. (2017, March)."A Selection Method for Black Box Regression Testing with a Statistically Defined Quality Level."In *Software Testing, Verification and Validation (ICST), 2017 IEEE International Conference on* (pp. 114-125). IEEE.

[20] Muzammal, M. (2016, December)."Test-Suite Prioritisation by Application Navigation Tree Mining."In *Frontiers of Information Technology (FIT), 2016 International Conference on* (pp. 205-210). IEEE.

[21] Haider, A. A., Nadeem, A., & Akram, S. (2016, December).'Safe regression test suite optimization: A review."In *Open Source Systems & Technologies (ICOSST), 2016 International Conference on* (pp. 7-12). IEEE.

[22] Liu, P. (2014, June)."An efficient reduction approach to test suite. In *Software Engineering,"Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on* (pp. 1-5). IEEE.

[23] Campos, J., & Abreu, R. (2013, April)."Encoding test requirements as constraints for test suite minimization." In *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on* (pp. 317-322). IEEE.

[24] Salehie, M., Li, S., Tahvildari, L., Dara, R., Li, S., & Moore, M. (2011, March)."Prioritizing requirements-based regression test cases: A goal-driven practice."In *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*(pp. 329-332). IEEE.

[25] Gonzalez-Sanchez, A., Abreu, R., Gross, H. G., & van Gemund, A. J. (2011, November)."Prioritizing tests for fault localization through ambiguity group reduction."In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering* (pp. 83-92). IEEE Computer Society.*RE, 2016 IEEE 27th International Symposium on* (pp. 47-58) IEEE.

[26] Bhuyar, P. R., & Gawande, A. D. (2015). Analysis using Non-Functional Static Testing Framework.

[27] Sahoo, R. K., Mohapatra, D. P., & Patra, M. R. (2016). A Firefly Algorithm Based Approach for Automated Generation and Optimization of Test Cases. *International Journal of Computer Sciences and Engineering*, *4*(8), 54-58.

Authors Profile

Ms. Arzu, pursued Bachelor of Technology in Computer Science & Engineering from Deenbandhu Chottu Ram University of Science and Technology, Murthal, Sonepat , Haryana India in year 2016. He pursuing M. Tech. from Deenbandhu Chottu Ram University OF Science and Technology Murthal , Haryana India in year 2018 . Her Research area includes Software Testing.

Mr. Sukhdip Singh, pursued Bachelor of Engg. in Computer Science & Engineering fromMaharishi Dayanand University Rohtak , Haryana India in year 1999. He pursued Ph.D. from Maharishi Dayanand University Rohtak , Haryana India in year 2013 . He is currently working as Associate Professor in the department of Computer Science & Engineering at Deenbandhu Chhotu Ram University of Science & Technology, Murthal , Sonipat Haryana India since 2002. He is a life member of Indian Society for Technical Education (ISTE) He has published more than 30 . Research papers in International Journals and Conferences of repute. His research areas includes Software testing, Software quality and computational intelligence . He has 18 years of teaching experience and 5 years of research experience.

