International Journal of Engineering & Technology, 7 (1.3) (2018) 95-99

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Enhancement of regression testing using genetic data generation and test case prioritization using m-ACO technique

S. K. Harikarthik [1]*, P. Ramanathan [2], V. Palanisamy [3]

[1] Assistant Professor, Dept. of Information Technology, INFO Institute of Engineering, Coimbatore, Tamilnadu, India
[2] Professor, Dept. of ECE, Madanapalle Institute of Technology & Science, Madanapalle, Chittoor District, Andhra Pradesh, India
[3] Principal [Retd], INFO Institute of Engineering, Coimbatore, Tamilnadu, India.
**Corresponding author E-mail: sk.harikarthik@gmail.com*

Abstract

The changes that occur during the software development process is rapid. Hence software has to undergo modification frequently. Due to this modification, the cost for testing increases due to repetitive retesting. This retesting process is called as the regression testing. Modification made in the single test case will make the side effect in all other related test cases. In order to overcome this problem all the test cases have to be retested again and again whenever the changes are incorporated in the software. But testing all the test cases is time consuming and will also increase the cost of testing. To address this problem, this work focuses on providing priority to the test cases. Test case which had more effect to changes is assigned with higher priority and the test case which had the less effect to changes is assigned lower priority. For test case prioritization, we employ m-ACO (Modified Ant colony optimization) method. Test case prioritization is done in two ways namely "Triangle classification problem" and "Quadratic Equation Problem". Flow of the data in the test case is done by Genetic Algorithm. This identifies the changed code in the program under test. It identifies both indirectly and directly affected def-use association in the modified part of the software by using forward walk algorithm and backward walk algorithm.

Keywords: *Regression Testing; Fault Coverage; Genetic Algorithm; Dataflow Testing; Def-Use Association.*

1. Introduction

Automation and need for software tools is drastically increasing in the current scenario. Most of the services are automated with computer software. Hence more skills are essential to architect the software and test it. But regressive testing process requires skill, time, effort and cost. Many research techniques are proposed to reduce the time and cost encountered in retesting process. Test case prioritization is an effective way to manage the test suite by appropriately scheduling the order of executing the test cases. The test suite prioritization is an important phase in regression testing, which rearranges the execution sequence of the test cases to detect the faults early with minimum effort [4] [5]. Kamna Solanki et al have reported test case prioritization technique by employing modified Ant colony optimization algorithm [1]. Their method alters the selection behaviour of the food sources by ants. After selection with prioritization, test cases are tested based on data flow in the modified test case. Dataflow analysis is done using genetic algorithm. It determines directly and indirectly affected def-use associations in the changed portion of the software by using backward walk and forward walk algorithm.

2. Literature review

Ant Colony Optimization (ACO) [20] is a meta-heuristic approach utilized for comprehending combinatorial streamlining issues. It has been effectively used to take care of numerous tricky optimizations issue. Simulated ants are effectively connected to yield a respectable amount of provisions, prompting universe class exhibitions for issues such as vehicle routing, quadratic assignment, scheduling, successive ordering, directing to web like networks and more.

Li et al [14] connected different meta-heuristics to experiment prioritization, mound climbing algorithm, hereditary algorithm and greedy algorithm. Rothermel et al. [3] attended to the issues identified with prioritization for expansive programming improvement situations. He settled down on an experimental study for analysing the effect about the measure of decrease in test suite and identified the shortcomings.

Walcott et al. [21] recommended algorithm with calculation greedy, additional greedy, 2-optimal, mound climbing and hereditary calculation for experiment streamlining. They acknowledged those taking into account three scope measurements independently (three solitary destination approaches): Average Percentage Block Coverage (APBC), Average Percentage Decision Coverage (APDC) and Average Percentage Coverage Statement (APCS). Their work reported that hereditary calculation will be superior to others for smaller projects and extra greedy, two-optimal algorithm are suited for large projects.

Singh et al. [1] exhibit a methodology of experiment prioritization issue dependent upon run time duration. The suggested ACO built calculation thinks of n ants, the place n specifies about test situations. The introductory vertebrate fossil science is decided haphazardly and the edges of the chart will be secured would haphazardly picked by those ants around ones Hosting greatest pheromone.

3. Objective

All paragraphs must be justified alignment. With justified alignment, both sides of the paragraph are straight.
On the premise about gap recognized from literature review, the targets for this work are:

- To review and understand the variety of presented technique for test case prioritization.
- To experiment Ant Colony Optimization technique for test case prioritization.
- To generate the test data by using Genetic algorithm
- Propose agenda for test case prioritization based on statement coverage using ACO algorithm and to test with data generated using Genetic Algorithm

4. Prioritization techniques

The m-ACO test case prioritizations are explained as follows [1].

a) m-ACO

m-ACO method is a technique employed to solve the test case prioritization problem. If there are 'n' test cases in the original test suite 'T' comprising of 'M' faults, we assure that number of modified ants in equal number of test cases for solution generation. Let T_i indicate the i^{th} test case in the test suite 'T'. The probability of test case sequences is indicated as S_{k1} for k-1 test cases selected after modification. First n modified ants are placed in alternative test sequence 'S_1'. Fig. 1 shows the m-ACO model for test case prioritization.

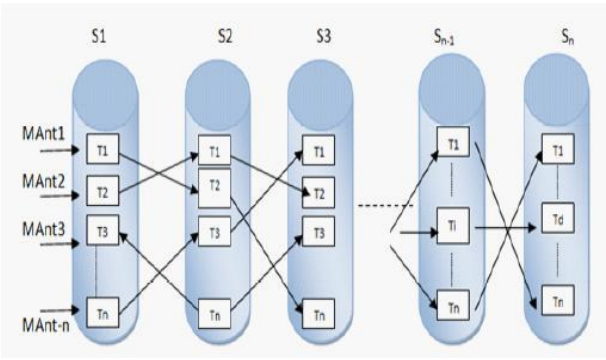

Fig. 1: M-ACO Model for Test Case Prioritization.

Fig. 2 describes how the real ants select the kind of food source which comes across randomly, while modified ant moves randomly and selects only those food source that are unique. Actually natural ants will always select the food source which is at shortest distance until it finishes food. Here by using m-ACO concept we are making the ants to select the foods that are of good quality located at the nearest site.
By adopting this concept, test case nodes are selected randomly by each of the modified ants in next sequence, until all faults are found by every modified ant.

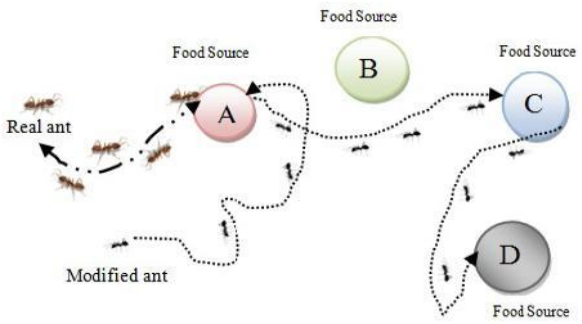

Fig. 2: Food Source Selection Behaviour of Natural and Modified Ants.

b) Evaluation based on experiment by using m-ACO technique

The proposed m-ACO technique for test case prioritization has been experimentally evaluated using two case studies reported in [5]. Two parameters that are considered for experimental evaluation are

- Average Percentage of Fault Detection [APFD]
- Percentage of Test suite Required for complete fault coverage [PTR]

APFD is used to identify the maximum fault detection rate by using a combination of test suites. The Average Percentage Fault detection is expressed as

$$APFD = 1-(\frac{(TF1+TF2+...+TFm)}{mn}+\frac{1}{2n})$$

Where 'T' is the test suites that need to be evaluated, 'm' represents the number of faults in an application under test and 'n' represents the total number of test cases in the test suite. TF_i represents the location of the i^{th} test case in test suite T that reveals the fault 'i'.
PTR is used to measure the effectiveness of the test suite prioritization technique. If PTR value is low then, it is a better prioritization technique.

$$PTR = \frac{NTCFC}{n} X\ 100$$

Where NTCFC represents the number of test cases needed for complete fault coverage.

c) Triangle classifier problem

Triangle classifier problem takes three sides of a triangle as an input and classifies the triangle as equilateral or scalene or isosceles triangle or not a triangle based on the input [1].
The sample test suite taken for triangle classifier problem comprises of 17 test cases in sequence {T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17} and 6 faults {f1,f2,f3,f4,f5,f6}. Table 1 refers the test suite for triangle problem with Execution Time (ET).

Table 1: Faults Detected by Test Suite for "Triangle Classifier Problem"

	f1	f2	f3	f4	f5	f6	ET
T1	*		*			*	5
T2	*	*	*			*	2
T3	*	*	*			*	3
T4	*	*	*	*			4
T5	*	*	*	*	*		5
T6	*	*	*			*	3
T7	*	*	*			*	6
T8	*		*			*	4
T9	*	*	*			*	5
T10	*	*	*			*	3
T11	*	*	*	*	*		8
T12	*	*	*			*	4
T13	*	*	*			*	6
T14	*		*			*	3
T15	*	*	*			*	2
T16	*	*	*			*	3
T17	*		*			*	5

After prioritization we attain the following result sequence as {T5-T2-T1-T3-T4-T6-T7-T17-T8-T9-T12-T15-T16-T10-T13-T14-T11} with an APFD value of 0.96 which is high compared to random sequence prioritization.

d) Quadratic Equation Problem

It is the famous software testing problems which takes the three variables of the equation as input and may produce the real roots or equal roots or imaginary roots. A simple test suite for quadratic equation problem comprises of 19 test cases in sequence of {T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15, T16, T17, T18, T19} and 9 faults named {f1, f2, f3, f4, f5, f6, f7, f8, f9}

Table 2: Fault Detected by Test Suite for "Quadratic Equation Problem"

	f1	f2	f3	f4	f5	f6	f7	f8	f9	ET
T1	*	*								3
T2			*					*	*	5
T3	*			*			*		*	2
T4	*			*	*				*	6
T5	*			*		*		*		3
T6	*		*			*			*	4
T7	*	*								2
T8										6
T9	*			*	*	*	*			4
T10	*		*		*	*		*		7
T11	*	*			*	*			*	3
T12	*			*	*	*		*		2
T13		*			*					7
T14		*	*							3
T15		*			*		*		*	5
T16	*			*	*		*		*	6
T17	*			*	*	*			*	3
T18	*		*		*	*		*		4
T19		*								1

The above mentioned test was executed using m-ACO, the generated result test sequence attained is as follows {T9,T7,T2,T1,T5,T4,T10,T3,T6,T15,T12,T11,T14,T16,T19,T13, T17,T18,T8} with an APFD value of 0.92.

e) Genetic Algorithm based test data generation.

The test-data generation using genetic algorithm is explained as follows:

i) Representation

The algorithm is represented by using a binary vector as a DNA and a Gene [chromosome] to represent the input variable x for the program.

ii) Initial population

The algorithm randomly generates POPSIZE m-bit strings for representing the initial population size. The POPSIZE value is determined by conducting the experiment. Each chromosome is converted as a K decimal value representing the values of T1....Tn where 'T' represents the test case.

iii) Evaluation Function

Algorithm evaluates the test cases {T1 ...Tn} by program execution with its input and it records the def-use paths in the program that are covered by this test case. The fitness value of each chromosome v1 (i=1...., POPSIZE)
is calculated as given below:

$$Fitness\_value\ (V_i) = \frac{No.of\ \ def\ -use\ path\ \ covered\ by\ V_i}{Total\ No.of\ \ def\ -use\ path}$$

The test case which is represented by the chromosome V_i is considered effective if its fitness value $(V_i) > 0$.

iv) Selection

The selection of the test case is done by prioritization of test case by using the two methods namely triangle classification problem and quadratic equation problem.

v) Recombination

Here the algorithm uses two operators namely crossover and mutation.

5. Experiment and result

In this section first prioritization is done based on the Triangle based problem and Quadratic equation problem. After prioritizing the test cases we will consider the following phases

i) Analysis & instrumentation phase.

ii) Affected def-use-pair generation phase.

iii) Test data generation phase.

iv) Experiment 1

In the first experiment the test cases are prioritized using triangle based problem with the data generation process done by using Genetic Algorithm. Here GA is used twice, the first time to get test data needed to cover all def-use pairs, and second time to get test data to cover the faulty def-use pairs. Table. 3, given below represents the input test cases, number of de-fuse pairs to be covered, number of generations and def-use coverage.

Table 3: Represents the Results of the Regression Testing Using GA for Test Cases Prioritized Using "Triangle Classifier Problem"

Test Case after prioritization	No. Of Def-use pairs to be covered		No of Generations		Def-use coverage %	
	All	Fault	All	Fault	All	Fault
T5	11	5	7	3	100%	100%
T2	7	4	3	1	55%	71%
T1	12	3	8	1	100%	100%
T3	10	4	6	3	100%	100%
T4	15	4	10	2	100%	100%
T6	13	4	9	1	60%	78%
T7	10	4	4	1	100%	100%
T17	7	3	5	1	100%	100%
T8	14	4	7	2	63%	69%
T9	12	4	5	2	60%	73%
T12	8	5	4	3	100%	100%
T15	21	4	2	1	100%	100%
T16	19	4	7	1	100%	100%
T10	24	3	8	1	100%	100%
T13	17	4	3	2	78%	81%
T14	14	4	4	2	100%	100%
T11	20	3	3	1	81%	98%

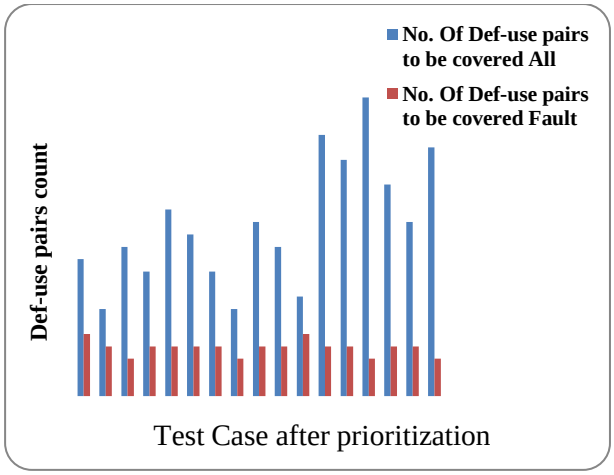

Fig. 3: Number of Def-Use Pairs in Each Test Cases.

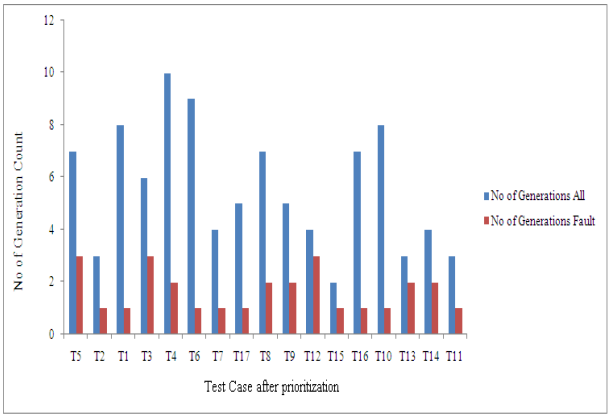

Fig. 4: Number of Required Generations.

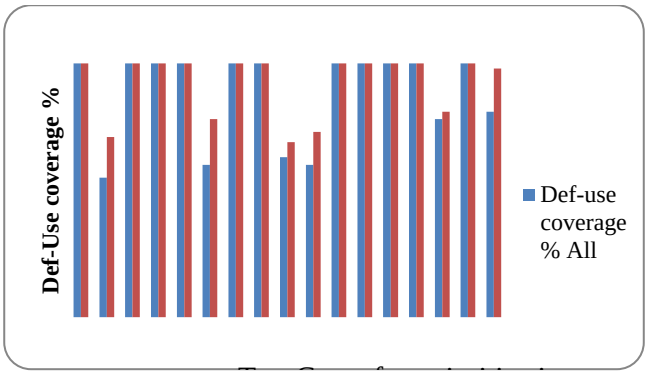

Fig. 5: Reduction Percentage of Def-Use and GA Generation.

v) Experiment 2

In the second experiment the test cases are prioritized using Quadratic equation problem with the data generation process done by using Genetic Algorithm (GA). Here also GA is used twice, the first time to get test data needed to cover all def-use pairs, and second time to get test data to cover the faulty def-use pairs. Table 4 given below represents the input test cases, number of defuse pairs to be coved, number of generations and def-use coverage.

Table 4: Represents the Results of the Regression Testing Using GA for Test Cases Prioritized Using "Quadratic Equation Problem".

Test Case after prioritization	No.Of Def-use pairs to be covered		No of Generation		Def-use coverage %	
	All	Fault	All	Fault	All	Fault
T9	20	4	5	1	100%	100%
T7	19	2	4	1	100%	100%
T2	33	5	13	1	100%	100%
T1	52	3	8	2	100%	100%
T5	21	3	7	1	100%	100%
T4	39	6	2	1	100%	100%
T10	17	7	17	1	100%	100%
T3	18	2	1	1	73%	81%
T6	13	4	7	2	100%	100%
T15	20	5	1	1	100%	100%
T12	9	2	4	2	74%	79%
T11	23	3	1	1	100%	100%
T14	44	3	1	4	100%	100%
T16	26	6	4	1	62%	71%
T19	13	1	3	3	100%	100%
T13	15	7	5	1	100%	100%
T17	19	3	4	1	100%	100%
T18	14	4	1	2	100%	100%
T8	21	6	2	3	100%	100%

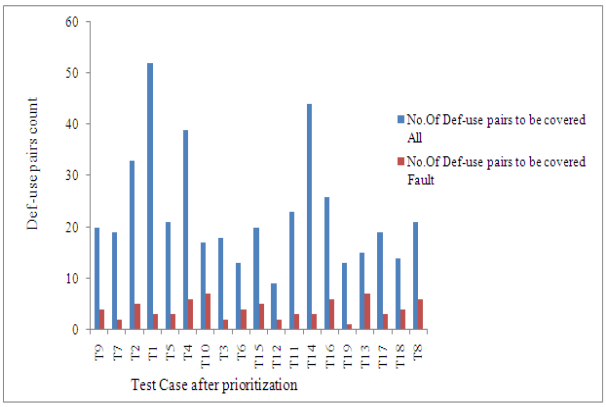

Fig. 6: No.of Def-Use Pairs in Each Test Cases.

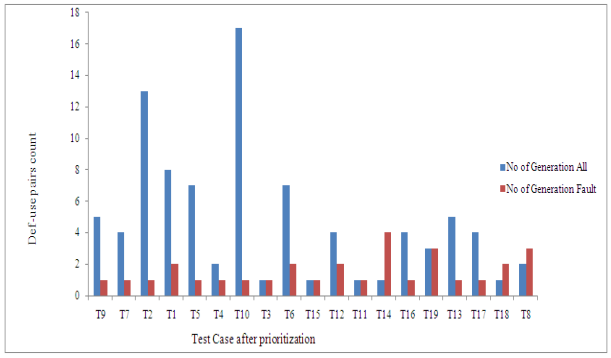

Fig. 7: No. of Required Generation.

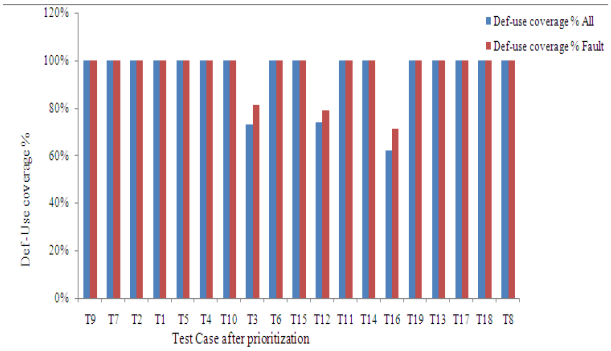

Fig. 8: Reduction Percentage of Def-Use and GA Generation.

From both the experiments we infer that the coverage percentage of fault def-use path by using GA is higher than the random path coverage. This is due to the fact that path coverage efficiency improves drastically because of test case prioritization.

6. Conclusion and future work

This paper presents an approach which employs test case prioritization with data generation for modified test case using GA. It can be used for both the object oriented programs and structural programs. First process prioritizes the test cases based on faults that occurred in the test case using triangle classifier problem method. After prioritization, fault occurred def-use paths are tested by using the test data generated by GA. Second process prioritizes the test cases based on faults that occurred in the test case using quadratic equation problem method. After prioritization, fault occurred def-use paths are tested by using the test data generated by GA. The results reveal that def-use coverage percentage has been increased drastically.

In future the experiment can be done by using clustering method which separates the wanted test cases from the unwanted ones before prioritization.

References

[1] Kamna Solanki, Yudhvir Sing and Sandeep Dalal, *Experimental Analysis of m-ACO Technique for Regression Testing*, Indian Journal of Science and Technology, August 2016, vol 9(30). https://doi.org/10.17485/ijst/2016/v9i30/86588.

[2] M.R.Girgis, A.S.Ghiduk and E.H. Abd-Elkawy., *An Approach for Enhancing Regression Testing Using Genetic Algorithm And Data Flow Analysis*, International Journal of Intelligent Computing and Information Science, April 2013, vol. 13

[3] Rothermal, G., Untch, R.H., Chu, C. and Harold, M., Test Case Prioritization, IEEE Transactions on Software Engineering, Vol. 27, No: 10, pp: 928-948, Oct. 2001.

[4] Chandu PMSS, Sasikala T. *Implementation of regression testing of test case prioritization,* Indian Journal of Science and Technology 2015 Apr; 8(S8):2903. https://doi.org/10.17485/ijst/2015/v8iS8/61922.

[5] Leung H, White L. *Insights into regression testing*. Proceedings of the IEEE International Conference on Software Maintenance; 1989 Oct. p. 60–9.

[6] Solanki K, Singh Y, Dalal S. *Test case prioritization: An approach based on modified Ant Colony Optimization*. Proceedings of IEEE International Conference on Computer, Communication and Control; Indore, India. 2015 Sept. Available at IEEE-xplore Digital Library.

[7] Beizer B. *Software Testing Techniques*. 2nd ed. India: Dreamtech Press; 2003.

[8] Catal C, Mishra D. *Test Case Prioritization: A systematic study. Software Quality*. Journal. 2013; 21(2):445–78. https://doi.org/10.1007/s11219-012-9181-z.

[9] Maheshwari V, Prasanna M. *Generation of test case using automation in software systems: A review*. Indian Journal of Science and Technology. 2015 Dec; 8(35):1–9. https://doi.org/10.17485/ijst/2015/v8i35/72881.

[10] Maheswari RU, JeyaMala D. *Combined genetic and simulated annealing approach for test case prioritization*. Indian Journal of Science and Technology. 2015 Dec; 8(35):1–5. https://doi.org/10.17485/ijst/2015/v8i35/81102.

[11] R. Gupta, M. J. Harrold, and M. Soffa, "*Program slicing-based regression testing techniques*". J. Software Testing Verification Reliability, Vol. 6, No. 2, pp. 83–111, June 1996. https://doi.org/10.1002/(SICI)1099-1689(199606)6:2<83::AID-STVR112>3.0.CO;2-9.

[12] G. Rothermel, and M. J. Harrold, "*Analyzing regression test selection Techniques*". IEEE Transactions on Software Engineering, Vol. 22, No. 8, pp. 529–551, August 1996. https://doi.org/10.1109/32.536955.

[13] A. Nanda, S. Mani, S. Sinha, M. J. Harrold, A. Orso, "*Regression Testing in the Presence of Non-code Changes*", 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation, pp. 21-30, 2011. https://doi.org/10.1109/ICST.2011.60.

[14] Li, Z., Harman, M., and Hierons, R. M., Search Algorithms for Regression Test Case Prioritization, IEEE Transactions on Software Engineering, Vol. 33, No: 4, April 2007. https://doi.org/10.1109/TSE.2007.38.

[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Mass., 1989.

[16] X. MA, B. Sheng, and C. Ye "*A Genetic Algorithm for Test-Suite Reduction*" Lecture Notes in Computer Science, Vol. 3756, pp. 253-262, 2005. https://doi.org/10.1007/11573937_28.

[17] S. Nachiyappan, A. Vimaladevi, and C. B. SelvaLakshmi, "*An Evolutionary Algorithm for Regression Test Suite Reduction*" Proceedings of the International Conference on Communication and Computational Intelligence, pp.503-508, 2010.

[18] Y. Zhang, J. Liu, Y. Cui, X. Hei, and M. Zhang, "*An Improved Quantum Genetic Algorithm for Test Suite Reduction*" 2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE), Vol. 2, pp. 149 – 153, 2011. https://doi.org/10.1109/CSAE.2011.5952443.

[19] Annamalai, R., and J. Srikanth. "Accessing the Data Efficiently using Prediction of Dynamic Data Algorithm." International Journal of Computer Applications, Vol. 116, No. 22, 2015.

[20] Aggarwal, K. K. and Singh, Y., Software Engineering, New Age International Publishers, 2005.

[21] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "*Design and evaluation of a wide-area event notification service*", ACM Transactions on Computing Systems, Vol. 19, pp. 332–383, August 2001. https://doi.org/10.1145/380749.380767.

[22] Z. Li, M. Harman, Hierons, and M. Robert, "*Search algorithms for regression test case prioritization*", IEEE Transactions on Software Engineering, vol. 33, No. 4, pp. 225–237, 2007. https://doi.org/10.1109/TSE.2007.38.

[23] Walcott, K. R., Soffa, M. L., Kapfhammer, G. M., Roos, R. S., Time-Aware Test suite Prioritization, Proceedings of the International Symposium on Software Testing and Analysis, pp:1-12, 2006.