International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-5, June 2019

An Ingenious KBC Approach for Test Suite Reduction to enhance Test Case Execution

Asha N and Prasanna Mani

Abstract: The cost and test resources needed for testing increases along with the test suite magnitude. To abode this magnitude snag an efficient systematic plan is proposed to derive a well-known nominal subgroup which will plentifully ceil the investigator's objectives. In the motive of corroborating the augmented version of software, it is fruitless of re-executing the entire fault-finding scenarios. An ingenious Knowledge-Based Collaborative (KBC) approach is formulated to face the test suite size problem, the minimal subset of test case is derived heuristically based on expertise testers' knowledge and the crucial fault-finding cases are ordered heuristically, so the most beneficial are executed first to enhance the test case execution. The lion's share of the fault checker is assessing the faults to heighten the system's working performance. The experimental study is made and the result is presented to illustrate the efficacy by incorporating the fault discovering cadent. Betterment credit achieved in ordering the fault-finding cases is shown by visualizing the comparative result of test case execution with ordered and disordered set of test cases based on fault espial.

Index Terms: Test Suite Reduction (TSR), Optimization, Average Percentage of Faults Detected (APFD), Expertise Knowledge

I. INTRODUCTION

The vibrant stage during SDLC to evaluate the functionality of mutated Software, to confirm the software has met the stakeholder's requirements and to ensure the quality product is made by identifying the defects and fixing the bugs. As the new version of software evolves the software grows which in turn it triggers to add new fault-finding scenarios in the test repository to test the latest modification made in augmented version of software. Lot more modifications made in software product leads to overabundance test domain.

More cost, testing resource has to be spent by elapsing more time to retest the augmented software. Investigators' substantial challenge is to test frequently with limited test resource. Bringing out an efficient test suite reduction approach is the only headlining practice for managing the test suite size problem; boosting the test result performance. The test suite reduction can be stated as Given: Let TTS be Total Test Suite that has m number of test cases {TC_1, TC_2, TC_3… TC_m} and TR are the Investigators' Requirements {IR_1, IR_2, IR_3… IR_n}, suffice the requirements for terrific test coverage. TTS = {TS_1, TS_2,..,TS_o } be the subgroup of TTS, each sub-group of suite is associated with any one of the requirements IR_i's, at least one test case TC_j of a TS_i satisfies R_i. Problem: Acquiring nominal subgroup of TTS whichever satisfies the investigators' imperative needs TS_i. The IR_i's can denote overall Requirements or the essential needs to investigate the modified part of software. The potential nominal subgroup of Test Suite termed as Hitting Set must accommodate a fault-finding case from each TS_i. The challenging part of work is achieving the maximum reduction with smallest subgroup. Only least number of researchers has noticed this problem to bring approximate solution for finding the smallest Representative Set, therefore an ingenious heuristic approach is formulated for deriving a progressive subgroup. Failure testing procedure retests the modified software to verify whether the bugs are rectified and the newly appended functionality created any new problem in the existing system. The failure must be verified regularly as a part of daily test cycle for the release that takes ample of months together to complete the test. But for small changes of software the failure test can be performed after the completion of functional test. The cost of Failure testing should be reduced by formulating a better approach which should also meet the time constrains in the execution environment. The other part of narration is coordinated as, Various Regression Testing Methodologies is embellished in section II, section III discusses the related work made on Test Suite Reduction and Section IV presents the proposed approach for Test Suite Reduction based on heuristics method. Section V depicts the experimental results. Finally Section VI summarizes the findings of the paper and the future work could be made.

Manuscript published on 30 June 2019.
* Correspondence Author (s)
Asha N , School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
Prasanna Mani, School of Information Technology and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India

II. REGRESSION TESTING METHODOLOGIES

Regression testing

As the new version of software is made, the modified part of code may cause error to the existing code, so the regression testing is conducted to re-run all the functional and non-functional tests to cross-verify that the previously constructed and tested software is defect free and performs well even after the modifications. Regression Testing levels are presented below:

A. Retest-all

The Retest-all technique aims to review entire software chunk under test to review glue integrity of the software. When the component C is modified to create C', the testers re-execute all the test cases to check for errors in C'. Though this testing is highly expensive it is must to ensure there are no errors in the modified code.

It's highly tedious to perform enormous number of testing, so to make this work ease the automated tool like assorted tools are incorporated to automate the testing and also an error monitoring tools can be used to recheck the missed part.

B. Regression Test Selection

Selective fault-finding scenarios are opted using opulent selection algorithm that covers all the test requirements, thus it is very effective in checking errors in unmodified part of code. Let TTS be total test suite TTS = {TC_1, TC_2, TC_3, TC_4, TC_5} and let TS' be the subset of TTS which contains selective test cases TS'= {TC_1, TC_2}, which could be feasible enough to manage the unmodified part and code by meeting the desired test requirements. Often used selection algorithms for test case selection are: Safe, Minimization, Random 25%, Random 50%, and Random 75%. Regression test selection comprises of three categories: Coverage techniques, Minimization techniques, Safe techniques.

C.Test Suite Reduction

Test Suite Reduction is the perfect pivotal key for dealing the magnitude snag. The main purpose of Test Suite Reduction is to pick the nominal subgroup which is equally potential to TTS. The coverage entities that are encountered for reducing the test elements are termed as investigators' requirements. The investigators' requirements suffice statement, conditional and branch coverage or system requirements. Also TSR technique eliminates the duplicated test elements from the Test Suite TS. It implicitly reduces the cost associated with the regression testing.

D.Test Case Prioritization

Test case prioritization techniques help in optimizing the fault-finding activity, highly prioritized crucial case is executed prior than the lower one. The test elements are organized according to the stakeholder's criteria like code coverage, branch, statement, condition coverage and fault detection rate. The test cases are not filtered but it is reordered in specified condition to improve the performance of testing

III. Literature Review

A Prioritized technique is framed heuristically. The proposed technique yielded optimal result when compared to that of random ordered and at most efficacy in performing the test execution within the specified time limit. The TCs are organized heuristically according to fault detecting capability. R. Kavitha et al. has formulated a contemporary method which identifies the fault detected rate and arranges the test cases on the impact of the fault. As a future work an efficient fault recognizing method can be framed to optimize the test execution. The tested data in the algorithm paves better solutions for test case execution. The testing was made for various coverage criterions: block, decision and statement. Many researches concentrated on test suite reduction, a significant analysis is conferred. Rothermel et al. made comparative study for technique formulated: random reduction, optimal prioritization, and no prioritization using the Siemens set programs. Among these the optimal prioritization is feasible which yields optimal result in limited test resources. Do and Rothermel et al. made practical coverage-based prioritization techniques to enhance the investigators' testing context. The method has enhanced the fault recognition rate. The ordered test cases has produced good outcome than the disordered test cases. Praveen Ranjan Srivastava et al. Introduced an ingenious practice to calculate standard flaws exposed. Fault discovering cadent, he has established the efficiency of the algorithm and presented. He has manipulated the usefulness of prioritized and no prioritized for discovering faults. Research made on deriving nominal subgroup with the motive of reducing the complications of investigators. Another research has identified potential test cases from each subset to obtain the hit set. The research paved way for fetching eventual suites with substantial decrease in the execution cost. Lot more research states the application of heuristic genetic concept which emulates the evolution process, which is extensively applied to verify the updated software version, specifically for software structural testing or for functional testing.

IV. Proposed work on Test Suite Reduction

Ingenious test design is made to review the developed software to spot out the bugs; many test case generation techniques are effectively emerging out that constitute the huge number of valuable scenarios which are highly capable enough to detect the flaws. As the evolution occurs still more test cases are appended to scrutiny the newly added features. Time consumed to run all these test cases cross the elasticity level of budget. So it's highly essential to concentrate on dealing with test suite magnitude snag. In regression testing, probability of executing the redundant test cases is high and its waste of time and effort. More attention has to be given to investigate the magnitude snag, to effectively obtain the nominal subgroup of TTS which is potential enough to suffice the Investigators' Test requirements (IR). To maintain the authentic capability in detecting the faults even though the test magnitude is diminished; the selected nominal subgroup is termed as Representative Set (RS). The TSR technique mainly spot out: (a) removing redundancy, (b) reducing test case execution cost and (c) abbreviating maintenance cost. Representative Set (RS) implicitly meets the Set Covering Problem. The Greed Heuristic G is very apt in solving the Set Covering Problem. Above all other techniques, the technique based on heuristic method is very popular for magnitude snag. With tacit knowledge the fault-finding cases can be categorized conferring to different levels of specifications. The selected nominal set is catalogued according to given criteria; in descending order from the most beneficial to the least beneficial to optimize the test case execution. In this paper another novel approach is framed based on heuristic and genetic method. The approach consist of various effective levels magnitude snag, levels are:

A.Review points awarded for test cases

In the evolutionary environment the probability of upgrading the software system is highly increasing. Due to the modifications made in the software, the testing activity has become a challenging task among the testers' environment. As the additional features that are added to the existing code, the modifications made to some part of existing code; the process of performing regression testing becomes very chaotic.

During the regression testing the test cases are reused. The Knowledge Repository concept supports in effectively reusing the test cases. Knowledge - based Test Suite Repository is maintained that systematically categorizes, retains and shares the Expertise Testers' knowledge. The performance and the characteristic nature of test case are reviewed by knowledgeable testers' which in turn is captured and stored in the database for the future usage of the corporate knowledge. The review point is awarded based on the capability of the fault-finding case at given circumstances. This review point is considered as the initial base for selecting the Initial Population Set (TS') from the TTS based on the given criteria. The chances of getting the hitting set will be high, because of following the expertise testers' knowledge in solving the problem.

B. Fitness Proportionate Selection

Fitness Proportionate Selection (FPS) process is applied on Initial Population Set (TS') to obtain the Representative Set (TS''). The FPS is an operator used in genetic algorithm for selecting potential chromosomes. If fi is tend to be fitness value of every test element in Initial Population, its probability of being selected is denoted by

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

p_i denotes the individual test elements probability
i denotes individual test case
f_i denotes the individuals' fitness value
N denotes Initial population

C. Select with Minimum Cost

In this approach of test suite reduction, apart from the review point of expertise testers', the individual's fitness value and the expenditure for test case is accounted for selecting test cases. The test case with merest execution cost is selected to get reduced set (TS^{RS1}). The approach is not only concerned with knowledge selection but with minimum cost also. The approach is framed economically by picking the potential fault-finding cases with minimum cost.

D. Perform OR Operation of Test Requirements

For the reduced set TS^{RS1} the TR (Test Requirement) matrix is constructed. It is a 0-1 two dimensional matrix with m number of test cases T = {TC_1, TC_2, TC_3… TC_m} and n number of requirements R = {R_1, R_2, R_3… R_n}, size of matrix is (m * n). Each row of the matrix determines the investigators requirements satisfied by individuals. If the requirements is satisfied by the test case it is denoted by 1, if not it is denoted by 0, Table. 1. The TR matrix is represented by

$$TR(i,j) = \begin{cases} 1 & \text{if } R_i \text{ is fulfilled by } TC_i \\ 0 & \text{if } R_i \text{ is not fulfilled by } TC_i \end{cases}$$

Using the TR matrix the second stage of reduced set is obtained TS^{RS2} by applying OR operations on the combination of TCs. Subgroup that suffice investigators' needs are opted. Condition to be noted here is, the requirements should be met by any one of the test case in combination set. The application of OR operation on sample combination TC set is represented below:
For the combination of test cases T = {TC_3, TC_8, TC_9}, fitness value will be:

TC_3 = { 1 1 0 0 0 0 0 1 0 0 }
TC_8 = { 0 0 1 1 0 0 1 0 0 1 }
TC_9 = { 0 0 0 0 1 1 0 0 1 0 }
OR { 1 1 1 1 1 1 1 1 1 1 }
Total Requirements = 10
Covered Requirements = 10

= (10/10) * 100 = 100%

Test Case No	Test Requirements (TR)									
	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
TC_1	1	1	0	0	1	0	0	0	1	1
TC_2	0	1	0	0	1	0	0	0	1	0
TC_3	1	1	0	0	0	0	0	1	0	0
TC_4	0	0	1	0	1	1	1	0	0	1
TC_5	1	0	0	1	0	1	0	1	0	1
TC_6	1	0	0	0	1	1	1	1	1	0
TC_7	0	0	0	0	1	1	0	0	0	1
TC_8	0	0	1	1	0	0	1	0	0	1
TC_9	0	0	0	0	1	1	0	0	1	0
TC_{10}	0	0	1	0	1	1	1	0	1	0

Table 1: Sample test case and the requirements satisfied by them

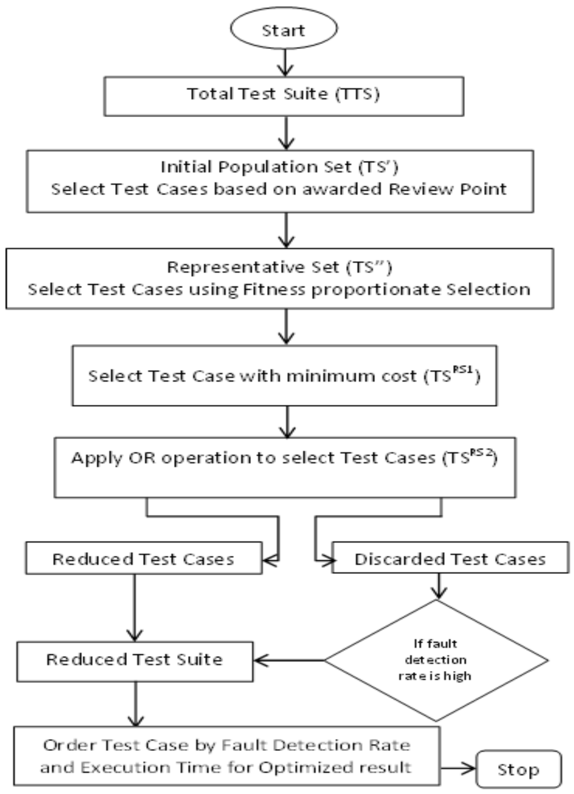
Start
Total Test Suite (TTS)
Initial Population Set (TS')
Select Test Cases based on awarded Review Point
Representative Set (TS'')
Select Test Cases using Fitness proportionate Selection
Select Test Case with minimum cost (TS^RS1)
Apply OR operation to select Test Cases (TS^RS2)
Reduced Test Cases
Discarded Test Cases
Reduced Test Suite
If fault detection rate is high
Order Test Case by Fault Detection Rate and Execution Time for Optimized result
Stop

Figure 1: Knowledge- Based Collaborative (KBC) Approach

For example, the combination set T = {TC_1, TC_2}, fitness value will be:

TC_1 = { 1 1 0 0 1 0 0 0 1 1 }
TC_2 = { 0 1 0 0 1 0 0 0 1 0 }
OR { 1 1 0 0 1 0 0 0 1 1 }
Total Requirements = 10
Covered Requirements = 5

= (5/10) * 100 = 50%

The sample test suite TS = {TC_3, TC_8, TC_9} obtained the fitness value of 100% by fulfilling all the requirements and the test suite TS = {TC_1, TC_2} obtained the fitness value of 50%. Hence the combination TS = {TC_3, TC_8, TC_9} is selected.
The set of

E. Retaining the Test Case

This level is specially meant for improving the potential capability to discover the faults thoroughly. Discarded test cases are re-considered in this stage. After all filtering process the test cases that are found to be highly potential in improving the quality are retained in the Suite.

F. Optimizing the Fault-Finding Execution

Main intention of final level of KBC is optimizing the execution result and boosting the capability of nominated subgroup. The potential test cases that are more beneficial and that has high fault detection rate are considered earlier than that has least rate. The fault detection (V_{Ti}) rate is computed by

V_{Ti} = Fault found / Time spent

According to the strength of fault found Value V_{Ti}, the TS^{RS2} subgroups elements are arranged in the descending manner, see algorithm 1. To quantify the result, APFD cadent is applied on the sample set of reduced test cases.

Algorithm 1: To optimize the test case execution result

```
Input: Fault Detected- FD, Reduced Test Suite- RTS and
Execution Time- ET.
Output: Re-ordered Reduced Test Suite- RTS'
1: Begin
2: Make the RTS' as null
3: For each Test Case $TC_i \in$ RTS do
4: Compute $V_{Ti}$ = FD/ET
5: End For
6: Sort test cases of RTS in descending order based on $V_{Ti}$ and
store it in RTS'
7: End
```

V. Experimental Result

The experimental result of KBC approach is checked on web application system (WAS). Table 1. Represents the sample of WAS considered for study. The experiment was conducted by accounting various attributes of test cases: Review Point, Fitness Value Point, Cost, and Fault Detection Rate. For the subjected WAS project the Test Suite of size 400 was extracted from the Test Suite Repository.

Test cases (TC)	Review Point	Fitness Value Point	Cost $	Fault Detection Rate
TC_1	4	3	79	60
TC_2	3	4	65	50
TC_3	5	5	70	80
TC_4	1	1	51	30
TC_5	2	1	40	40
TC_6	3	2	43	50
TC_7	4	3	42	30
TC_8	5	5	70	80
TC_9	5	5	70	90
TC_{10}	1	2	34	20

Table 2: Test Cases: Review point, Fitness Value point, Cost and Fault Detection Rate

TTS is the Total Test Suite taken into account for experimental study.

TTS = {TC_1, TC_2, TC_3,..., TC_a}

As a first level of abstraction the Initial Population Set (IPS) is obtained from the TTS via considering the review point awarded by the expertize testers' knowledge.

TS' = {TC_1, TC_2, TC_3,..., TC_b}

In the second level of abstraction the Representative Set (RS) is derived from the IPS by applying Fitness Proportionate Selection method.

TS'' = {TC_1, TC_2, TC_3,..., TC_c}

In the third level of abstraction the Reduced Set is obtained further by accounting the Minimum Cost of individuals.
The test case with minimum execution cost is selected.

TS^{RS1} = {TC_1, TC_2, TC_3,..., TC_d}

In the fourth level of abstraction the Reduced Set is acquired by applying OR operation on the combination of test cases. Replication in meeting investigators' requirements should be avoided.

TS^{RS2} = {TC_1, TC_2, TC_3,..., TC_e}

In the fifth level of abstraction, the test cases that hold high fault detection rate is retained back to revamp the execution result.

RTS = {TC_1, TC_2, TC_3,..., TC_f}

Above all levels of filtering, finally the shortlisted test elements are arranged according to fault ratio, which optimizes execution result. Table 3. presents the fault matrix with execution time. The Fault ratio found is measured using the APFD cadent; this metric clearly states the percentage of faults found by the nominal set. T is the test suite that contains n test cases and F is the set of m faults detected through any of the test cases. TF_i is the position of test case in the organized test cases which detects the stated flaw first. Using all these data the APFD is computed by the formula:

$$APFD = 1 - \frac{TF_1 + TF_2 + TF_3 + \ldots\ldots\ldots\ldots + TF_m}{nm} + \frac{1}{2n}$$

Test cases / Fault	TC_1	TC_2	TC_3	TC_4	TC_5	TC_6	TC_7	TC_8	TC_9	TC_{10}
F_1						*			*	
F_2							*			*
F_3	*		*			*			*	
F_4		*		*		*		*		
F_5								*		*
F_6			*		*				*	
F_7	*		*		*					
F_8										*
F_9							*			
F_{10}									*	
Number of Faults	2	1	3	1	2	3	2	2	4	3
Time (ms)	6	9	16	12	8	14	11	13	9	14

Table 3: Test Suite's: Fault Detected and Execution time

To demonstrate the fault detection rate the sample of test cases of Web Application System is considered with 10 Test Cases and 10 Faults which is represented in Table. 3. Also the Table. 3 insinuate faults rectified by chosen test cases and total execution time of every test elements.

To optimize the result the nominated test elements are arranged according to specified criterion. Since fault ratio is considered to be an important criteria to be fulfilled the test case are organized based on the fault detection rate in KBC approach. The fault detection rate is computed as:
Fault Detection Rate:

(V_{Ti}) = Number of Fault Detected / Execution Time

The Fault Detection Rate is enumerated with V_{Ti} formula, the calculated V_{Ti} value is displayed in Table. 4

V_{Ti}	Value
V_{T1}	0.33
V_{T2}	0.11
V_{T3}	0.18
V_{T4}	0.08
V_{T5}	0.25
V_{T6}	0.21
V_{T7}	0.18
V_{T8}	0.15
V_{T9}	0.44
V_{T10}	0.21

Table 4: Rate of Fault Found

The TS^{RS2} elements are arranged in higher order of V_{Ti} value. The most potential test cases are picked first to improve the beneficial of the approach. Using values displayed in Table 4. , the test cases are arranged as follows:
T_9, T_1, T_5, T_6, T_{10}, T_3, T_7, T_8, T_2, T_4
To prove that the ordered test cases yields optimized result than that of the disordered test case. The APFD is calculated for the ordered and disordered elements of TS^{RS2}.

Value of APFD for well-organized TCs:

$$APFD = 1 - \frac{4+2+4+1+5+3+3+5+2+4}{10 * 10} + \frac{1}{2 * 10} = 0.72$$

Value of APFD for tangled TCs:

$$APFD = 1 - \frac{6+7+1+2+8+3+1+10+7+9}{10 * 10} + \frac{1}{2 * 10} = 0.51$$

Evidently 72% of faults were spotted because of running elements of TS^{RS2} in well- organized structure, whereas the disordered set produces 51%, thus the test case execution is optimized. Figure 2. illustrates the graphical representation of ratio of fault diagnosed via well-organized elements of TS^{RS2} and Figure 3. Show the disordered set. The shaded area beneath the curve specifies the fault ratio identified in the interim of testing stage.

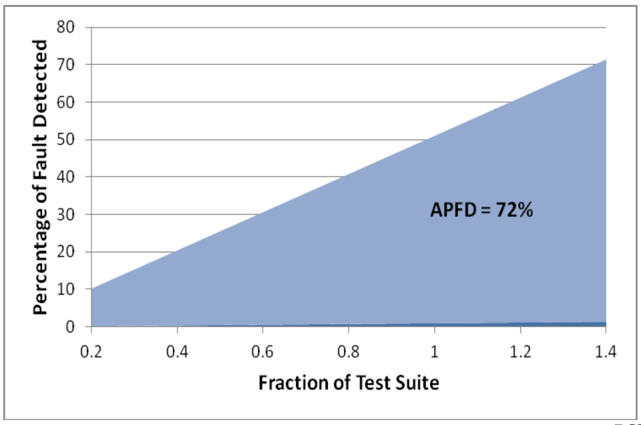

Figure 2: Fault ratio for well-organized elements of TS^{RS2}

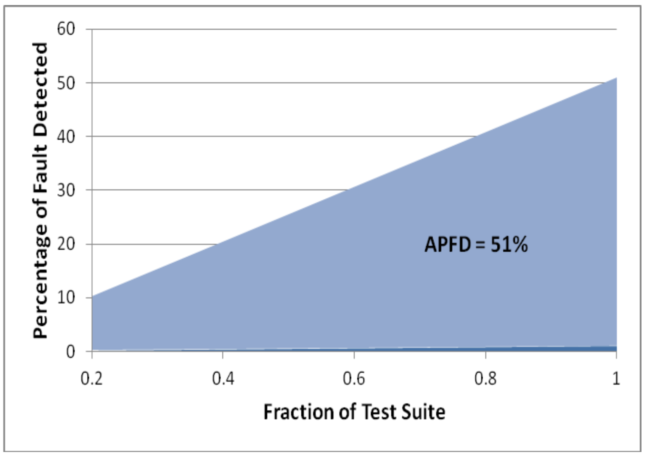

Figure 3: Fault ratio for tangled elements of TS^{RS2}

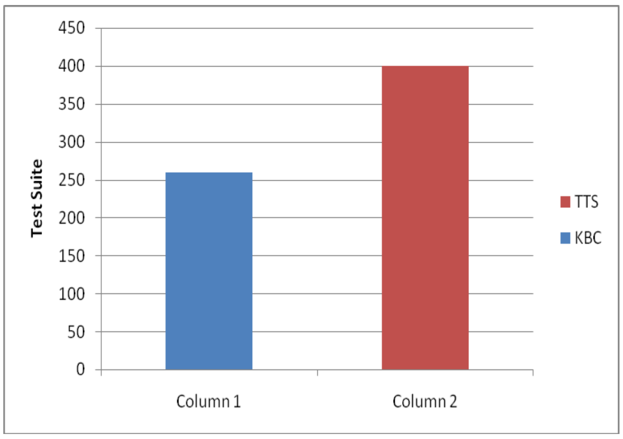

Figure 4: Comparative result of KBC approach

The efficiency of the KBC approach is graphically represented in the Figure 4. The total test suite size considered for testing is 400 and by applying the KBC approach heuristically the magnitude is reduced to the extent of 263 from the choate of 400. Thereby 65.75% percentage of reduction is achieved which implicitly reduces the cost and the test resources required for testing the Web Application System.

VI. CONCLUSION

The major problem that arises in the testing environment is test suite size problem. The significant nature of software system is its endless state of meeting the changes in it. As the alteration rises in software chunks; implicitly new test elements are appended in the test suite. It is highly challenging to face the risk associated with magnitude snag of test elements. Cost, test resources, time needed for executing the test cases drastically increases. Many researchers have been made on regression testing methodologies to solve the size problem. The chaotic nature of testing environment can be handled effectively by applying the heuristic methods. A novel approach named as Knowledge-Based Collaborative (KBC) approach is proffered and represented.

It comprises of different reduction levels magnitude snag. Along the reduction measure an optimization technique is also appended finally to bring an optimal solution for enhancing test result. The APFD cadent metric is incorporated to measure the fault ratio. Fault ratio is optimized from 51% to 72% by organizing the test cases based of fault detection rate. The KBC approach has achieved 65.75% percentage of reduction and yielded a better solution for magnitude snag. For future enhance the KBC can be refined by considering multiple criteria for bringing the optimal solution. The proposed approach can be compared with the advanced technique to present the comparative result for better optimization.

REFERENCES

1. M.J. Harrold, R. Gupta, and M.L. Soffa, “A Methodology for Controlling the Size of a Test Suite,” ACM Trans. Software Eng. And Methodology, vol. 2, no. 3, pp. 270-285, July 1993.
2. G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. Proceedings of the International Conference on Software Maintenance, pages 34-43, November 1998.
3. G. Rothermel, M. J. Harrold, J. von Ronne, and C. Hong. Empirical studies of test-suite reduction. Journal of Software Testing, Verification, and Reliability, V. 12, no. 4, December, 2002.
4. W. Eric Wong, Joseph R. Horgan, Saul London, Aditya P. Mathur. Effect of test set minimization on fault detection effectiveness. Proceedings of the 17th Internationa Conference on Software Engineering, p.41-50, 1995, Seattle, Washington, United States.
5. W. E. Wong, J.R. Horgan, A. P. Mathur, and A. Pasquini. Test set size minimization and fault detection effectiveness: A case study in a space application. Proceedings of the 21st Annual International Computer Software and Applications Conference, pages 522-528, August 1997.
6. G. Hunt and D. Brubacher. Detours: binary interception of Win32 functions. Proceedings of the 3rd USENIX Windows NT Symposium, pp. 135-143. Seattle, WA, July 1999.
7. A.S.Syed Fiaz, N.Asha, D.Sumathi & A.S.Syed Navaz “Data Visualization: Enhancing Big Data More Adaptable and Valuable” February – 2016, International Journal of Applied Engineering Research, Vol No - 11, Issue No - 4, pp.–2801-2804.
8. A.S.Syed Navaz & Dr.G.M. Kadhar Nawaz “Flow Based Layer Selection Algorithm for Data Collection in Tree Structure Wireless Sensor Networks” March – 2016, International Journal of Applied Engineering Research, Vol No - 11, Issue No - 5, pp.–3359-3363.
9. Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum, Cost cognizant Test Case Prioritization, 2006
10. Wang, Xingya, et al. "Cost-effective testing based fault localization with distance based test-suite reduction." Science China Information Sciences 60.9 (2017): 092112.
11. Gladston, Angelin, et al. "Test suite reduction using hgs based heuristic approach." Computing and Informatics 34.5 (2016): 1113-1132.
12. Lin, Jun-Wei, et al. "Nemo: multi-criteria test-suite minimization with integer nonlinear programming." 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018.
13. R. Kavitha and N. Sureshkumar, (2011) , “Factors Oriented Test Case Prioritization Technique in Regression Testing”. European Journal of Scientific Research.
14. S.Elbaum, A.Malishevsky, and G.Rothermel, (2002), Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering.
15. Asha N and Prasanna Mani ”Customized services to generate test suits for testing custom software application system based on knowledge reuse”, Journal of Advanced Research in Dynamical and Control Systems, 01-Special Issue, 2017. Pp-14-20.
16. Asha.N, A.S.Syed Navaz, J.Jayashree & J.Vijayashree “RFID Based Automated Gate Security System”, November – 2018, ARPN Journal of Engineering and Applied Sciences, Vol No - 13 , Issue No - 22, pp.–8901 – 8906.
17. Praveen Ranjan Srivastava, (2008) “Test Case Prioritization”, Journal of Theoretical and Applied Information Technology IEEE.
18. M. Roper, CAST with GAs (Genetic Algorithms) - Automatic Test Data Generation via. Evolutionary Computation, IEE Colloquium on Computer Aided Software Testing Tools, digest no. 96/096, April, 1996.
19. A.S.Syed Navaz & Dr.G.M. Kadhar Nawaz “Layer Orient Time Domain Density Estimation Technique Based Channel Assignment in Tree Structure Wireless Sensor Networks for Fast Data Collection” June - 2016, International Journal of Engineering and Technology, Vol No - 8, Issue No - 3, pp.–1506-1512.
20. A.S.Syed Fiaz, K.S.Guruprakash & A.S.Syed Navaz “Prediction of Best Cloud Service Provider using the QoS Ranking Framework” January – 2018, International Journal of Engineering & Technology, Vol No - 7, Issue No -1.1, pp.– 486-488.
21. Pargas, R., Harrold, M., Peck, R., Test data generation using genetic algorithms. Software Testing Verification & Reliability, vol. 9, no. 4, pp. 263-282,1999.
22. Michael, C., McGraw, G., Schatz, M., Generating Software Test Data by Evolution, IEEE Transactions On Software Engineering, 27(12),pp. 1085-1110, December 2001.
23. Wegener, J.; Baresel, A. and Sthamer, H., Evolutionary Test Environment for Automatic Structural Testing. Information and Software Technology, Special Issue devoted to the Application of Metaheuristic Algorithms to Problems in Software Engineering, vol. 43, pp. 841 - 854 (2001).
24. A. Watkins, The Automatic Generation of Software. Test Data using Genetic Algorithms, Proceedings of the Fourth Software Quality Conference, 2: 300-309, Dundee,. Scotland, July, 1995.
25. Borgelt, K., Software Test Data Generation From aGenetic Algorithm, Industrial Applications of Genetic Algorithms, CRC Press 1998.
26. A.S.Syed Navaz, P.Jayalakshmi, N.Asha. “Optimization of Real-Time Video Over 3G Wireless Networks” September – 2015, International Journal of Applied Engineering Research, Vol No - 10, Issue No - 18, pp. 39724 – 39730.
27. JC.Lin and PU. Yeh, Automatic Test Data. Generation for Path Testing using GAs, Information Sciences, 131: 47-64, 2001.reduction technique.
28. M.J. Harrold, R. Gupta, and M.L. Soffa, “A Methodology for Controlling the Size of a Test Suite,” ACM Trans. Software Eng. And Methodology, vol. 2, no. 3, pp. 270-285, July 1993.
29. D. Jeffrey and N. Gupta, “Improving Fault Detection Capability by Selectively Retaining Test Cases During Test Suite Reduction,” IEEE Trans. on Software Engineering, Vol. 33, No. 2, pp. 108-123, February 2007.
30. Ma, X.y., He, Z.f., Sheng, B.k., Ye, C.q.: “A genetic algorithm for test-suite reduction”. In: Proc. the International Conference on Systems, Man and Cybernetics, pp. 133–139, October 2005.
31. Chu-Ti Lin, Kai-Wei Tang, Cheng-Ding Chen, and Gregory M. Kapfhammer. “Reducing the Cost of Regression Testing by Identifying Irreplaceable Test Cases” . In Proc. Of the 6th ICGEC ’12.
32. Y Zhang, J Liu, Y Cui, X Hei , ”An improved quantum genetic algorithm for test suite reduction “,IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011.
33. A.S. Syed Navaz, N. Asha & Kuna Venkateswararao “Face Recognition and Detection with New Image Enhancement Techniques” December – 2018, Journal of Advanced Research in Dynamical and Control Systems, Vol No -10 , Issue No – 14 (Special Issue), pp.– 760-764.
34. A.S.Syed Fiaz, M. Usha and J. Akilandeswari “A Brokerage Service Model for QoS support in Inter-Cloud Environment“,March 2013, International Journal of Information and Computation Technology, Vol.3, No.3, pp 257-260,
35. M. Usha, J. Akilandeswari and A.S.Syed Fiaz “An efficient QoS framework for Cloud Brokerage Services”, Dec. 2012, International Symposium on Cloud and Service Computing, pp: 76-79, 17-18, IEEE Xplore.
36. Sudhir Kumar Mohapatra, Srinivas Prasad “Minimizing Test Cases to Reduce the Cost of Regression Testing“, June 2014, International Conference on Computing for Sustainable Global Development (INDIACom), pp 505-509, IEEE Xplore.
37. A.S.Syed Navaz, N.Asha & D.Sumathi “Energy Efficient Consumption for Quality Based Sleep Scheduling in Wireless Sensor Networks” March - 2017, ARPN Journal of Engineering and Applied Sciences, Vol No - 12, Issue No - 5, pp.–1494-1498.
38. Asha N and Prasanna Mani ” Applying knowledge based heuristic method for efficient test automation”, Journal of Advanced Research in Dynamical and Control Systems, Issue: 12-Special Issue, 2018. Pp- 1216-1223.
39. Asha N and Prasanna Mani ” Knowledge-based acceptance test driven agile approach for quality software development”, International Journal of Recent Technology and Engineering, Dec- 2048, Vol – 7, Issue: 4S2, 2018. Pp- 196-202.

40. A.S.Syed Navaz, Asha.N, Vanmathi Chandrasekaran & J.Jayashree "Resourceful Investigate Encryption Method Using Data Hunt in Mobile Cloud Service", August – 2018, ARPN Journal of Engineering and Applied Sciences, Vol No - 13, Issue No - 15, pp.–4543-4549.
41. Asha N and Prasanna Mani "Test Suite Reduction based on knowledge Reuse: An Adaptive Elitism Based Intellect approach (AEBI) using Clustering Technique" International Journal of Innovative Technology and Exploring Engineering, Volume-8 Issue-6, April 2019. Pp.-1566-1572.