ISSN 2347 - 3983
Volume 8. No. 5, May 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter104852020.pdf
https://doi.org/10.30534/ijeter/2020/104852020

Fault Aware Test Case Prioritization in Regression Testing using Genetic Algorithm

Priyanka Paygude[1], Dr. Shashank D. Joshi[2], Dr. Manjusha Joshi[3]
[1,2]Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune, India, pspaygude@bvucoep.edu.in , sdjoshi@bvucoep.edu.in, [3]SKNCOE, Vadgaon, Pune, India, manjushajoshi1@gmail.com

ABSTRACT

In software testing practice, regression testing is significant type of testing, which is responsible for stability, quality, reliability and functionality of existing application even after doing the modifications in the application. Test case prioritization (TCP) is one of the proved effective techniques of regression testing that improves defect detection rate by scheduling execution of the test cases (TCs). Scheduling the execution of all TCs is very complex and time consuming task and thus needs to introduce the use of optimization algorithms. This paper implements a genetic optimization algorithm (GA) to improve the TCP technique by ordering the TCs with goal of maximum fault detection by minimal execution of TCs. The effectiveness of implemented GA optimization technique is measured using average percentage of fault detection (APFD) metric. We analyzed implemented GA approach to examine its effect on outcome by changing its vital parameters such as crossover, mutation and convergence criteria with the aim of increasing rate of fault detection. This experiment is evaluated on public dataset with more than 1000 TCs. We tend to compare our work with random search prioritization and hill climbing optimization algorithms. This carried out experimental outcome clearly depict that GA outperforms better than compared algorithms in solving TCP problem by improving the performance of regression testing.

Key words: APFD, genetic algorithm, regression testing, test case prioritization

1. INTRODUCTION

Generally, system in its development and maintenance phase continuously undergoes modifications and thus system must be tested before and after the changes are merged into the main development part. Software undergoes continuous up gradations due to ever changing customer requirements. As software grows, the database of test cases grows exponentially, that makes testing more complex. If the software is not tested aptly then there is high risk [1, 2] of defects retention in the system under test (SUT). Regression testing plays a significant role in approving the quality and reliability of the system undergoing continuous up gradation. When software system is modified for its functionalities, it is expected to retest all test cases for existing and newly added functionalities before the release of product to the customer. Regression testing is the final step that verifies and works as a quality measure to confirm that changes made in the system has not affected the previous correctly working functionalities and the system is still stable as far as working is concerned. This confirmation can be achieved by re executing all the TCs. But, re-executing all TCs before each release is a costly and time-consuming task as test suite grows exponentially with addition or modification in system. This can be controlled and made possible by introducing prioritization in execution ordering of the TCs. Test case prioritization (TCP) is widely adopted technique in software industry to reduce efforts and effective execution of regression testing [3-6]. TCP orders the TCs of test suites by calculating priority based on test suitability criteria such as early fault detection, critical fault coverage, code coverage, requirement coverage etc.

In literature survey of TCP, there are many approaches proposed, implemented and validated for addressing TCP problem for effective regression testing. Many search optimization algorithms such as genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), simulated annealing (SA) and history-based approach are widely research areas in TCP problem. It has been found from the literature study that GA optimization technique is mostly implemented and proved effective for TCP. But limitation we observe from literature that many research works have generated their own test suite manually and that too with less than 100 TCs.

In this paper, we have applied and implemented GA for solving the TCP problem in regression testing using history based testing information. We have not only implemented GA, but also tried to inspect the effect on outcome of GA on changing its vital parameters- crossover, mutation. We compared and analyzed the outcome of variations in parameters on 4 large datasets. Evaluation of experiment is carried out on the publicly available dataset bigfaultmatrix having 1000+ test cases and 37 faults [27].For comparison, we have considered Average Percentage of Fault Detection (APFD) [4] metric which depicts the rate of fault detection on execution of test suite. We compared the GA technique with other algorithmic approaches, such as Random Search, Hill Climbing (internal and external swap).

This paper is organized as follows. Section 2 discusses the literature work in TCP using GA. Session 3 explains regression testing and TCP concept. Implemented GA, its algorithm and flow is explained in section 4. Section 5

discusses experimental evaluation. Section 6 describes comparison of results of GA approach with other algorithms. we have concluded our work in section 7.

2. LITERATURE SURVEY

This session surveys and discusses the previous papers published for solving the TCP problem using optimization algorithms [5]. Use of optimization techniques is widely researched area for TCP due to its successful outcomes. We have focused our survey study on the use of GA and its hybrid combinations for addressing TCP problem. GA along with many hybrid approaches are proposed and evaluated in literature such as GA with Particle Swarm Optimization (PSO), GA with Ant Colony Optimization (ACO), GA with Greedy Algorithm, GA with Simulated Annealing (SA) [8,17] etc.

The research work [10] proposed a multi-objective-based GA approach for ordering the test cases using code coverage parameters such as total statement coverage, total fault exposing weight of each TC and total mutant coverage by the TC. The technique is evaluated on classic problem of triangle classifier with limited manually generated TCs and fault matrix. The proposed GA-PSO hybrid technique [11] works in two parts, first – the initial population is iterated for fitness function yielding optimized set of population using genetic evolutionary concept. In second part, initial results from GA are given as input to PSO, where problem of converging at local optima is resolved using global best. The results of proposed techniques are significant but are based on static test case- fault dataset. The work [12] presented a new prioritization algorithm with the aim of uplifting the rate of fault detection using history of fault detection by test cases and severity of faults. Authors have compared their proposed method with existing work [13]. The results evaluated of static dataset of [13] prove the efficiency of proposed technique. The hybrid combination of adaptive approach with GA is suggested in [14], achieves the 100% statement coverage. In first step, adaptive approach will calculate fault detection capability of each test case and high rank TCs are executed prior to other TCs. In second step, leftover TCs are given as input to GA for generating optimized sequence. In [15] authors have designed a Component-Based Software testing prioritization framework using GA with Java decoding technique to enhance the software quality by detecting faults at the early stage of software development. In [16], authors have developed an automation tool which is a hybrid combination of Genetic Algorithm (GA) and Simulated Annealing (SA) algorithm for TCP. Execution time and fault detection rate are used as input parameters for calculating fitness value. This tool evaluated on static test case fault matrix shows promising results compared to traditional test suite execution order.

It has been observed through carried literature survey that very few papers have referred public datasets or tested approach on real projects. Almost 90% of the studied papers tried their approach on manually generated test suite that too with less than 100 TCs. With observed limitations from related work, we set the goal of implementing GA, assessing its performance on big public datasets and observe effect on GA by changing its implementation parameters.

3. REGRESSION TESTING AND TCP

Regression testing is the process of discovering and confirming that modifications added in the application have not adversely affected the existing working functionalities and application still work as expected. Regression testing ensures this by re-executing all or partial test cases. It assures that new changes have not penetrated any defects in the previously tested and working application [18-21]. Figure 1 shows, how scope and efforts for regression testing increases exponentially as system develops and undergoes modifications. So, after doing changes or adding new functionality in the system, the system is prone to introduce new defects, thus regression testing is needed. Here testing is carried out for modified and existing functionalities keeping more attention on impacted functionalities.

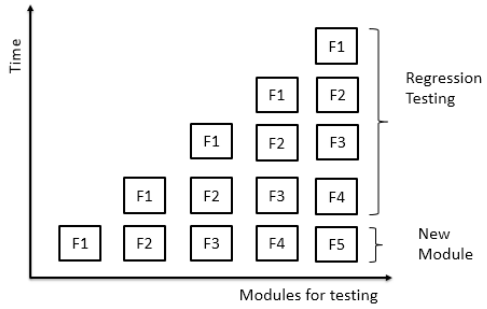

Figure 1: Scope and efforts for regression testing

Regression Test Selection (RTS), Regression Test Minimization (RTM) and Test Case Prioritization (TCP) [3] are the approaches considered for regression testing of the application. RTS and RTM select the test cases from test suite based on experience of tester and thus by elimination TCs, it raise the possibility of defects retention in the system. TCP is the most adopted technique, which works without elimination of test cases it schedule the execution order of test cases by achieving certain objectives such as early fault detection, maximum fault coverage, core functionalities [22-24]. In broad way, TCP techniques are classified into coverage based, fault based, requirement based, risk based etc. Test case prioritization approach is to derive the execution order of the test cases based on various criteria or properties such as statement / branch/ code coverage data, fault detection ability, execution time.

4. IMPLEMENTATION OF GENETIC ALGORITHM

Genetic Algorithm is an evolutionary algorithm used to solve the combinatorial optimization problems which are usually computationally very expensive to solve. The base of genetic algorithm is the biological concept of evolution and the survival of the fittest [26]. This algorithm is much more powerful and efficient than exhaustive search algorithm as it provides good and robust solution rated against fitness criteria. It is used for solving optimization problems where objective is to find optimum value i.e. finding maximum

(global maxima) or minimum (global minima) value. Genetic algorithm is a population based probabilistic search and optimization technique, which works based on mechanism of natural genetics and natural evaluation. Fitness function evaluates how much a given solution is close to optimality.

We have implemented GA to solve TCP problem, for which we have considered population of chromosome where each chromosome is a possible sequence of test cases selected from test suite. This TCP problem is ordering of TCs and is discrete in nature. The process begins with initial population generation. We have used permutation encoding to encode the chromosome. Here, we have considered population size as 200 and is randomly generated from the test suite. Each chromosome from population evaluates for fitness score by applying fitness function i.e. Average Percentage of Fault Detection (APFD). The fitness value i.e. APFD is the number of faults detected by chromosome by running minimal test suite. The termination criteria considered for stopping the iterations is the number of generations. While termination or converge criteria is not matched, chromosomes undergoes tournament selection. Randomly any two individual chromosomes are selected, which further undergoes for single point crossover where point of crossover is chosen randomly. Mutation phase will randomly mutate the bits of chromosome to generate better children for next generation. We have run the experiment for variations of crossover and mutation probability rates. The detail results are discussed in next section. The survivor selection is performed based on fitness score above probability value 0.75. Generated offspring continuously iterate till the convergence criteria is not met. Figure 2 shows flowchart of GA implementation.

Algorithm:

Input: Test Cases from BigFaultMatrix Test suite, TS
Output: Ordered test suite, TS'

```
START
  initialize population from BigFaultMatrix data file
  Encoding of population
  find fitness of population (APFD)
  while (termination criteria is not reached) do
    parent tournament selection
    crossover with probability pc
    mutation with probability pm
    decode and fitness calculation ft
    survivor selection
    find best
  return best
END
```

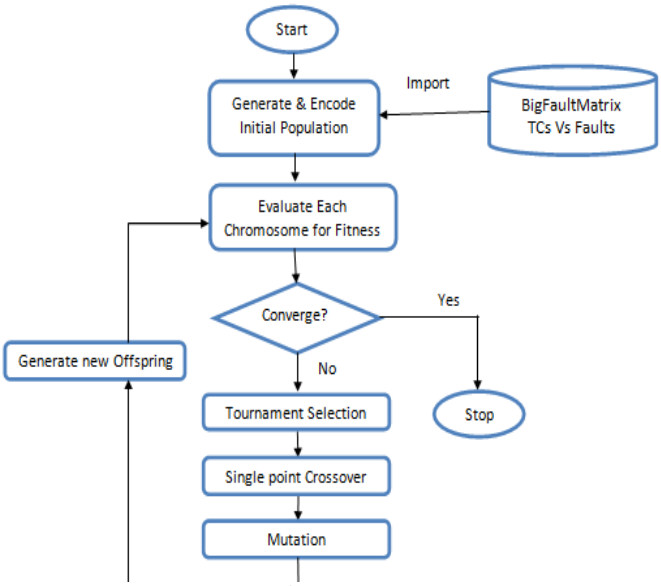

Figure 2: Flowchart of proposed GA approach

5. EXPERIMENTAL EVALUATION

We have executed the experiment on the public dataset available on Github [27] by the title BigFaultMatrix. This is a big dataset which is a fault matrix consisting of 1000 test cases and 38 faults made available for software testing research purpose. The data stored in matrix is of binary format, where 1 indicates faults covered and 0 indicates faults not covered by respective test case. We are considering this as our 1st case study. We did variations in the dataset by randomly flipping bits and generated 3 more case studies. APFD metric is considered as fitness function and is applied to compute the score of fault detection rate [28]. APFD measures the total number of faults detected by the prioritized test case order. We have executed GA on each case study for different variations of crossover rate, mutation rate and number of generations to converge as mentioned in the table 1. The results are plotted using bar chart for deliberating variations. In graph, the x-axis represents the variation rate and the y-axis represents respective APFD values calculated.

Table 1:GA Parameters Variations

Parameters	**Variations**			
Crossover Rate	0.6	0.7	0.8	0.9
Mutation Rate	0.1	0.2	0.3	0.4
Number of Generations	3	6	9	12

The initial parameters for Genetic algorithm are set for population size 200 and chromosome size 50. With these parameters, GA was executed on four case studies with different crossover rate i.e. 0.6, 0.7, 0.8 and 0.9. Following figure 3 to 5 are the results of case studies with variations in crossover rate. Figure 6 is the result of APFD values against the variation in mutation rates for case study 1 by keeping fixed crossover rate 0.5. Number of generations is the parameter considered as convergence criteria to stop GA iterations. Figures 7 to 10 shows the variations of converging

GA iterations with different mutation rates. The result depicts that, GA results best with parameter values as crossover rate 0.8 and mutation rate 0.2. Also, from the experiment we can conclude that for considered dataset, there is almost no change after 12th number of generations. So, this we have considered as convergence criteria for GA iterations.

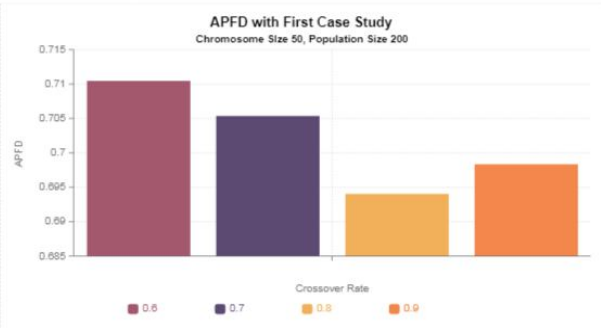

Figure 3: APFD measure with crossover variations- case study 1

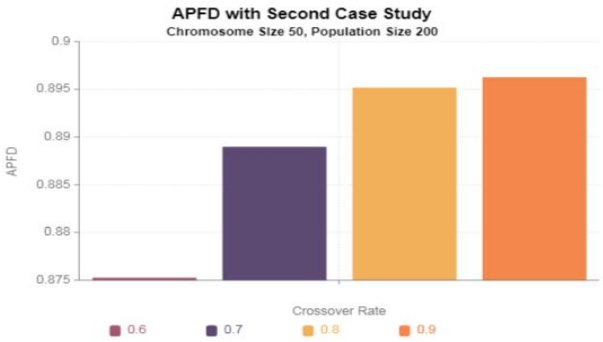

Figure 4:APFD measure with crossover variations- case study 2

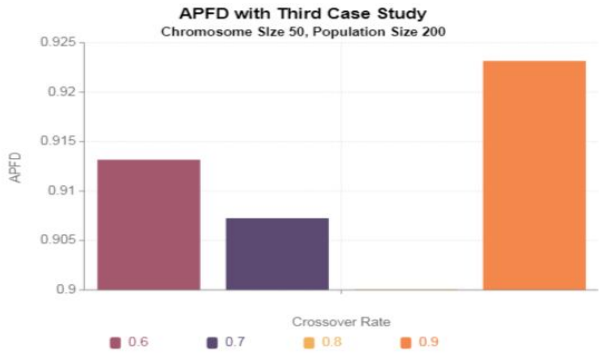

Figure 5: APFD measure with crossover variations- case study 3

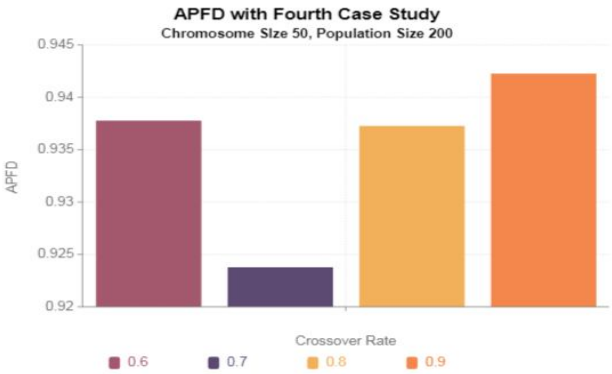

Figure 6: APFD measure with crossover variations- case study 4

Figure 7: APFD measure with mutation variations- case study 1

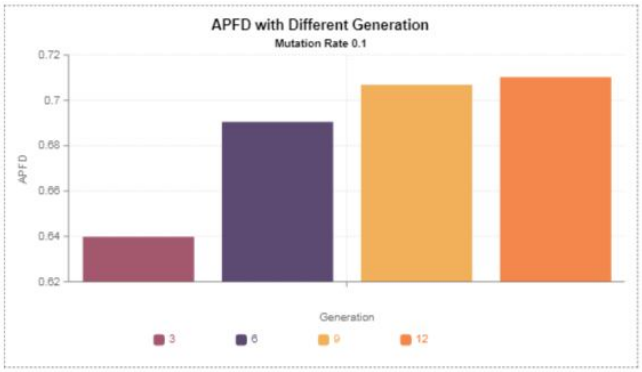

Figure 8: Variation in no. of generation - case study 1

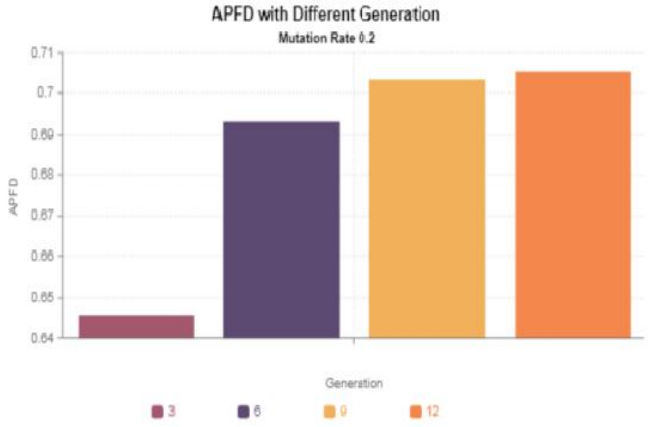

Figure 9: Variation in no. of generation - case study 2

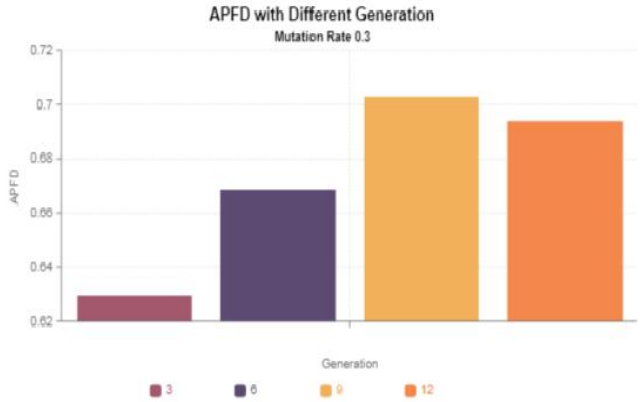

Figure 10: Variation in no. of generation - case study 3

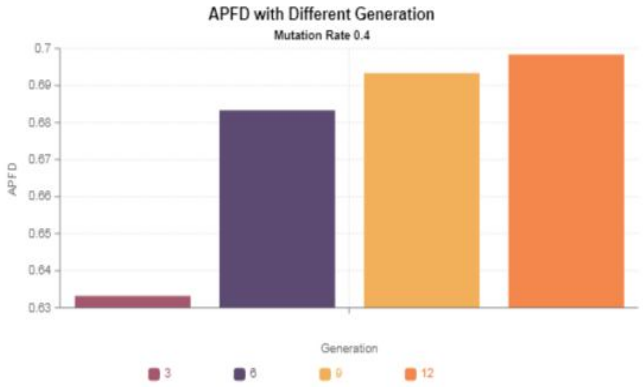

Figure 11: Variation in no. of generation - case study 4

6. COMPARISON AND RESULT DISCUSSION

From previous section results the best value of APFD was chosen to compare the GA technique with other algorithmic approaches, such as Random Search, Hill Climbing (internal and external swap). Following are the parametric values for the experiment we carried out so that we can carry out the algorithmic comparison over APFD values.

The proposed and compared techniques are assessed for showing its effectiveness in scheduling the test cases in order to maximize rate of fault detection (APFD). We compared and analyzed the outcomes of our proposed algorithm with other algorithm on the same dataset and found that GA outperforms the compared techniques. Following table 2 summarizes our results.

Table 2: Comparative Results of algorithm

Name of Algorithm	APFD
Random Search	0.5425
HC Internal Swap	0.5258
HC External Swap	0.5307
Genetic Algorithm	0.7053

From above table 2 it shows that Genetic algorithm gives maximum APFD i.e. 70% and outperforms compared to other algorithms like Random Search (54%), HC Internal Swap (52%) and HC External Swap (53%). The same had been depicted in following figure 12.

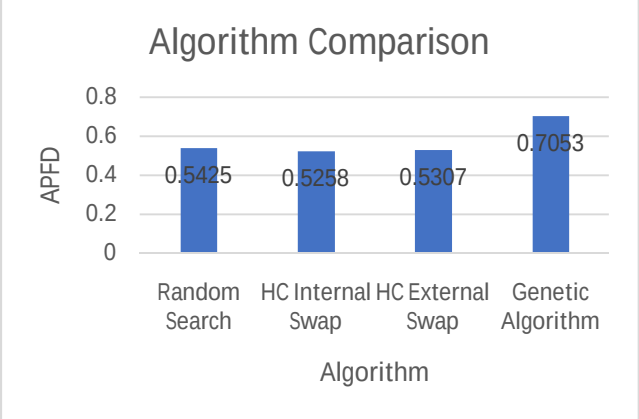

Figure 12: Comparison of algorithms based on APFD measure

7. CONCLUSION

We have conducted an empirical study with the motive of examining genetic algorithm performance for scheduling of test cases with the goal of early fault detection and to check effectiveness of GA on variations of its vital parameters. Previous empirical studies have proved that genetic algorithm works better in ordering the test cases but were tested on test suite having test cases not more 100 and many times generated fault matrix is manual. Here, we have implemented and tested the GA approach on TCP problem on more than 1000 TCs.

The first motive of study was to check effect GA for its performance on changing its vital parameters- crossover rate, mutation rate and number of generations for stopping GA iteration. We examine these variations on four big fault matrices as input datasets. The experimental outcome shows that GA performs more impressive when crossover rate set to 0.8 and rate of mutation set to 0.2.

Comparing GA performance with other algorithms was second motive of study. This study results clearly depicts GA efficiency is much better than compared random and hill climbing algorithms.

In future, we have planned to execute proposed algorithm on real time projects. Here, we have only considered only fault detection rate as single objective to calculate APFD. Future scope of this research work is to do the hybridization of GA with other suitable algorithm and also to adapt it for multi objective TCP parameters in order to improve GA effectiveness.

REFERENCES

[1] Seong Ho Sung, Pattan Zinna Khan, **Quantitative and Qualitative Approach for IT Risk Assessment**, Asia-pacific Journal of Convergent Research Interchange, SoCoRI, Vol.1, No.1, pp. 29-35, March 2015
https://doi.org/10.21742/apjcri.2015.03.04

[2] Gowtham Kumar Pullagujju, **Risk Utilization in Quantitative Approach**, Asia-pacific Journal of Convergent Research Interchange, SoCoRI, Vol.1, No.1, pp. 21-27, March 2015
https://doi.org/10.21742/apjcri.2015.03.03

[3] Yoo, Shin, and Mark Harman. **Regression testing minimization, selection and prioritization: a survey**. Software Testing, Verification and Reliability vol. 22, no. 2, pp. 67-120, 2012.

[4] Han Moi Sim, D. Sai Teja Reddy, **Survey on Test Case Prioritization and Measuring Test Cases Using APFD**, Asia-pacific Journal of Convergent Research Interchange, SoCoRI, Vol.1, No.2, pp. 11-17, June 2015.
https://doi.org/10.21742/apjcri.2015.06.02

[5] Singh, Yogesh, et al. **Systematic literature review on regression test prioritization techniques** *Informatica* Vol.36, no. 4, 2012.

[6] G. Chandrika, **Study on Software Reliability and Reliability Testing**, Asia-pacific Journal of Convergent Research Interchange, SoCoRI, Vol.1, No.1, pp. 7-20, March 2015

[7] Dong Ju Kim, P. Lakshmi Manjusha, **Assessment of Risks in Management Factors**, Asia-pacific Journal of Convergent Research Interchange, SoCoRI, Vol.1, no.2, , pp. 1-10, June 2015
https://doi.org/10.21742/apjcri.2015.06.01

[8] Su Min Shin, Sk. Uroosa, **Predicting Software Reliability Using Particle SWARM Optimization Technique**, Asia-pacific Journal of Convergent Research Interchange, SoCoRI, Vol.1, No.3, pp. 17-30, September 2015

[9] Amol K. Kadam, S.D. Joshi, Debnath Bhattacharyya and Hye-Jin Kim.**Diagnosis of Software using Testing Time and Testing Coverage.** International Journal of Hybrid Information Technology. Vol. 9. No. 9. Sep. 2016 https://doi.org/10.14257/ijhit.2016.9.9.08

[10] Mishra, Deepti Bala, Namita Panda, Rajashree Mishra, and Arup Abhinna Acharya. **Total fault exposing potential based test case prioritization using genetic algorithm.** International Journal of Information Technology, pp. 1-5, 2018.

[11] Saraswat, Pavi, and Abhishek Singhal. **A hybrid approach for test case prioritization and optimization using meta-heuristics techniques**. In Information Processing (IICIP), 2016 1st India International Conference on, pp. 1-6. IEEE, 2016.

[12] Nayak, Soumen, Chiranjeev Kumar, and Sachin Tripathi. **Effectiveness of prioritization of test cases based on Faults**. In Recent Advances in Information Technology (RAIT), 2016 3rd International Conference on, pp. 657-662. IEEE, 2016.

[13] R. Kavitha and N. Sureshkumar. **Test case prioritization for regression testing based on severity of fault.** International Journal of Computer Science and Engineering (IJCSE), vol. 02, no. 05, pp. 1462-1466, 2010.

[14] Walia, Rajanroop, and Harpreet K. Bajaj. **Performance Analysis of Hybrid Approach Comprising Genetic Algorithm and Adaptive Approach on Test Case Prioritization**. International Journal of Computer Applications, Vol. 155, no. 8, 2016.
https://doi.org/10.5120/ijca2016912403

[15] Mahajan, Surendra, Shashank D. Joshi, and V. Khanaa. **ComponentBased Software System Test Case Prioritization with Genetic AlgorithmDecoding Technique using Java Platform**. International Conference on Computing Communication Control and Automation (ICCUBEA). IEEE, pp. 847-851, 2015.
https://doi.org/10.1109/ICCUBEA.2015.169

[16] Maheswari, R. Uma, and D. Jeya Mala. **Combined genetic and simulated annealing approach for test case prioritization**. Indian Journal of Science and Technology vol. 8, no. 35, 2015

[17] Jin Wang, Yu Gao, Wei Liu, Arun Kumar Sangaiah, Hye-Jin Kim, **An Improved Routing Schema with Special Clustering using PSO Algorithm for Heterogeneous Wireless Sensor Network**, Sensors, vol.19, no.3, Feb. 2019

[18] Catal, Cagatay, and Deepti Mishra. **Test case prioritization: a systematic mapping study.** Software Quality Journal, Vol. 21, no.3, pp: 445-478, 2013
https://doi.org/10.1007/s11219-012-9181-z

[19] Kumar, Amit, and Karambir Singh. **A Literature Survey on test case prioritization**. Compusoft, Vol. 3, no. 5, 2014.

[20] Kiran, R. Surya. **A literature survey on TCP-test case prioritization using the RT-regression techniques.** Global Journal of Research In Engineering, 2015.

[21] Hao, Dan, Lu Zhang, and Hong Mei. **Test-case prioritization: achievements and challenges.** Frontiers of Computer Science, Vol. 10, no. 5 pp. 769-777, 2016.

[22] Khatibsyarbini, M., Isa, M. A., Jawawi, D. N., & Tumeng, R. **Test case prioritization approaches in regression testing: A systematic literature review.** Information and Software Technology, Vol. 93, pp. 74-93, 2018.

[23] Mukherjee, Rajendrani, and K. Sridhar Patnaik. **A survey on different approaches for software test case prioritization.** Journal of King Saud University-Computer and Information Sciences, 2018.
https://doi.org/10.1016/j.jksuci.2018.09.005

[24] Saraswat, Pavi, Abhishek Singhal, and Abhay Bansal. **A Review of Test Case Prioritization and Optimization Techniques**. Software Engineering. Springer, Singapore, pp. 507-516, 2019.

[25] Jin Wang, Jiayi Cao, R. Simon Sherratt, Jong Hyuk Park, **An improved ant colony optimization-based approach with mobile sink for wireless sensor networks**, Journal of Supercomputing,Vol. 74, No.12, pp.6633-6645, Dec. 2018.
https://doi.org/10.1007/s11227-017-2115-6

[26] J.H. Holland. **Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology.** Control, and Artificial Intelligence, U Michigan Press, 1975

[27] https://github.com/dathpo/Test_Case_Prioritisation_-_Genetic_Algorithm/blob/master/bigfaultmatrix.txt

[28] Elbaum, S.G., Malishevsky, A.G., Rothermel, G., **Test case prioritization: a family of empirical studies**. IEEE Transaction on Software Engineering, vol. 28, no.2, pp.159–182, 2002.
https://doi.org/10.1109/32.988497