ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse175952020.pdf
https://doi.org/10.30534/ijatcse/2020/175952020

Test Case Prioritization Using Cat Swarm Optimization

Richa Vats[1] , Arvind Kumar[1]
[1]SRM University Delhi-NCR, Sonepat (Haryana), 131029, India
ritzi1606@gmail.com, k.arvind33@gmail.com

ABSTRACT

Regression testing is one of time taking and expensive task. It is an indispensable part of SDLC process. The main task of regression testing is to scuttleall test cases of a given test suite. But it is time consuming and tedious task and also needed more efforts and time. The effort can be described in term of human work. One of the solutions for aforementioned task is to automate the entire process through prioritizing algorithm. The aim of thee algorithms is to determine optimal test-cases for regression testing from a given test suite. The other objective of these algorithmsis also addressed the time and effort issues of regression testing through achievinghigher faults detection. Hence, in this work, a new technique based on CSO techniqueis implemented to prioritize test cases. Prior to apply the CSO method for prioritizing test-cases, some amendmentsare inculcated in CSO algorithm to achieve optimum results in terms of fault detection. The experiment demonstratesthe applicability of CSO based prioritization technique for attaining effective results especially prioritizing task.

Key words: Prioritization Algorithm, Fault Detection, Clustering, Software Engineering, CSO Algorithm, Regression Testing

1. INTRODUCTION

As in past few decades, there is tremendous growth in software field. Large numbers of software are developed for automation, day to day operations, healthcare, manufacturing etc. But, prior to delivery of the software's, it requires testing of the developed software. It is very time consuming and panic task. Hence, it can be defined as to evaluate the capabilities like behavior, failure etc., of a software under different circumstances and also evaluates its features using predefined criteria's. Moreover, this process also monitors the development of software. Further, the testing is also done to validate the software and detect the errors. A study showed that this process requires sixty percent of cost and also effort of SDLC. Different test cases are generated to evaluate test data and meet the testing criteria. In software development, testing can be described as an important activity for validating the performance of software. The aim of testing is to detect the unusual behavior of software through test-cases and also identify errors in software. This can be done through sequence of inputs and produced expected output. Further, due to dynamic and competitive environment, the requirements of user are changed frequently. In turn, testing process become tedious, time-consuming and complex task as per developer and tester perspectives and more challenging. It is also observed that frequently changes in requirement also lead to frequent change in test-cases and it can overload the tester as to test new test-cases as well as store available test-cases. Because in future, any updation can be done in software, thentester will be able to reuse test-cases. This process can include large numbers of subroutines, functions and statements of an application system. It is also noticed that most of software development organizations believed that software is independently designed, and also having better security and testing abilities [1]. Hence, testing can be described as to determine the faults in software application using test-cases for improving the quality of software. However, test-cases cannot be design easily, optimal test-cases takes lot of time and effort. Further, these test-cases also contain several subroutines which are initially described and output of these test cases should be predictable. The uncertainty is also associated with designed test-cases as program will pass the specific test-case accurately or not. It can be described through well intended, deliberated, scheduled and prioritized process[2]. Further, this process also highlights the reason for failure of program.The primary task of prioritization algorithms is to investigate the role of test-cases for validating the software application. These algorithms also determine the optimal subset of test-cases from a test suite in a hope that this subset of test-cases fully investigate the software application instead for executing the entire test-cases presented in test suite. Finally, a tester can schedule test-cases in such a manner that its traverse maximum code in minimum time and also reducing cost factor. Large number of prioritization algorithms are reported in literature for test suite optimization. These algorithms can be worked with bugs removal, test-case execution etc. This work introduces a CSO based algorithm for prioritizing the test-cases. CSO algorithm inspired through the behavior of cats and applied in diverse field [3-10]. Its stated that CSO based prioritization algorithm obtains more accurate results for test-cases prioritization.

2. RELATED WORKS

This section describes recent works on TCP and it is listed below.

Harrold et al. [11]selected the test-cases characteristics set for developing a test suite minimization method. These selected test cases givesame coverage like complete test suite with reduced cost of regression testing. A new method to assess regression testing in terms of completeness, correctness, effectiveness, and generalization is presented in [12]. In this work, author has developed some criteria to weather a regression test either safe or unsafe. To choose the optimal test-cases, some techniques are presented in [13-14].The other techniques for prioritizing test-casesare presented in [15]. The performance of these techniques is evaluated using fault detection rate.To improve the rate of fault detection using based on TCPis reported in [16]. This work introduces version-specific TCP concept. This concept enhances the fault rate detection in regression testing. Moreover, some version specific prioritization techniques are also reported to prioritize the test cases for a test suite [17-18]. In continuation of their work, another technique based on the magnitude of cost effectiveness is also presented in [19]. A cost effective prioritization method is also reported in [20]. A prioritization technique for object-oriented programming language is reported and this technique gained wide popularity among programmer [21]. A regression testing based on black box testingis developed by Qu et al. [22]. This technique can assemble all faulty test cases together. Further, the priorities associated with technique are adjusted as per results of test-cases dynamically. An approach based on historical valuefor TCP is reported in [23]. This technique is used for computing the severity of faults. Further, historical value can be used for computing the cost to a cost-cognizant TCP based on historical value. In software testing, minimization and prioritization of test-cases are important aspects. So, a study on these aspects is presented in [24]. Bajwa and Kaur developed an adaptive approach for test cases prioritization based on GA [25].It can speed up the scheduling of test cases. To improve the coverage of test cases prioritization, an immune based genetic algorithm is reported in [26], in the proposed approach, an immune operator is incorporated in genetic algorithm to overcome the low convergence problem. It is noted that IGA give better results thangenetic algorithm. A hybrid approach based on genetic algorithm and SA is reported for TCP [27]. This approach can reduce cost as well as enhance fault rate. Tulasiraman and Kalimuthu developed a cognizant cost and history basedTCP approach [28]. The proposed approach computes fault rate and cost based on historical information of test-cases. Moreover, artificial immune system algorithm is also applied to find the effective test cases. A multiobjective search-based regression TCP approach is presented in [29]. It is amalgamationof epistasis theory and ant colony optimization algorithm (ACO). The epistasis theory is used to update the pheromone strategy of ACO algorithm. To enhance the effectiveness of TCP, Chen et al. [30], presented an adaptive random sequence approach. The proposed approach consists of two clustering algorithm such as K-means and K-medoid.The simulation results stated that the proposed approach enhances earlier detection of fault rate. To detect the faults earlier, a fuzzy TPOSIS technique is reported to prioritize test-cases [31]. In this approach, fuzzy principles are used for decision making. A risk based prioritization approach is reported for test cases [32]. In this work, fuzzy expert system is developed to accurate detection of risks or faults. Noguchi et al. [33] developed a frame work for TCP using ant colony optimization algorithm. Jiang and Chan presented local beam search basedtechnique for effective TCP [34]. The proposed approach is validated using four benchmarks test cases datasets and gives better results than greedy and genetic algorithms.Prioritizing the test cases based on total coverage, Konsaard and Ramingwong applied a modified genetic algorithm for TCP [35]. A greedy based prioritization approach is reported for optimizing the TCP problem [36]. The proposed approach consists of exploration strategy and multi level coverage model to capture the bugs. The MOGA is reported for TCPto overcome regression testing cost [37]. In this work, a mechanism based on orthogonal design and evolution is incorporated in multi objective GA. It is seen that DIV-GA is more capable than other algorithms.To optimize the test cases in time constrained environment, panwar et al. [38] presented a hybrid approach by combining CS and modified ACO algorithm for obtaining optimized test cases. A Bayesian based clustering approach is presented to prioritize test cases [39]. In this work, two java projects are considered to identify the mutated faults. The performance of the work is compared with greedy approach and BNA techniques. It is stated that Bayesian based clustering gives promising results. To detect faults with minimum time and earlier, Tulasiraman et al. [40] presented pareato and clonal selection algorithm based multi-objective approach for TCP. It is noticed that proposed multi objective approach scheduled the test cases optimally and earlier. Suri and Singhal presented ACO based technique for regression testing and prioritization [41]. Further, it is seen that a time bounded constraint is incorporated in proposed approach to determine optimal test cases. Results confirm that ACO based technique is one of effective technique for TCP.

3. PROPOSED APPROACH PRIORITIZING BASED ON CLUSTERING

3.1 Motivation

The motivation of this research is to develop CSO based prioritization algorithm for the identification of optimum test-cases from a test suite. Generally, test suite comprises of large number of test-cases and execute each test-case is a time-consuming task. In turn, testing effort will be increased. Furthermore, test-cases are characterized on the basis of attribute types and each test-case having some common characteristics. Hence, to reduce the testing effort and also accelerate the testing process, a CSO based prioritization algorithm is proposed. The CSO based prioritization algorithm works in two steps. In first step, test-cases presented in test suite are separated into k-clusters based on similarity or dissimilarity measure. The test-cases occurs similarity placed in one cluster, whereas, dissimilar test-cases put into different clusters. So, a cluster contains more than one test-case that are similar in nature and occurs heterogeneity with other clusters. The second step towards to selection of test-cases from clusters in random order. For testing purpose. The main of work is to detect maximum faults with respect to minimum test-cases.

3.2 Cat Swarm Optimization

It is arecent technique that can be adopted for searching best solution through mimic the behavior of cats. This algorithm represents the potential solution for the optimization problems in terms of cats position. Further, optimum solution is refined through two modes of CSO algorithm, these modes can explain in terms of seeking mode and tracing mode. Seeking mode denotes the moving characteristics of cats, whereas, tracing mode corresponds to hunting capabilities of cats. A flag value is used to determine the modes of cat i.e. current presence of cat in modes. Two amendments are introduced in CSO algorithms to make it more powerful.

1. A newsearch mechanism is devised for maintaining local and global searches of CSO.
2. A conditional operator is applied to determine whether the algorithm executes in exploration phase or exploitation phase. This conditional operator measures the current cost of gbest position of cat is compared with previous cost. If, current cost is less than previous cost then algorithm execute in exploration phase. If significant difference occurs between the cost function, then algorithm executes in exploitation phase.
3. To improve the convergence rate, an inertia weight function (w) is also used with searching mechanism of CSO algorithm.

3.3 Pseudo Code of CSO Based Prioritization Algorithm

The steps of CSO based prioritization algorithm are listed as

```
1. Load test-cases from test suite and initialized the different user defined parameters of CSO based prioritization algorithm such as population of cats, flag etc.
2. Randomly evaluate the catspositions and determine velocities of cats.
3. Determine the similarity and dissimilarity between test-cases and clusters; and grouped test-cases into different clusters through similarity and dissimilarity measure.
4. Evaluate the fitness ($Fit_G(X)$) of cats and store the positions of cats into variable $X_G$ and denote best position of cats using ($X_G$).
5. Check the flag value, If flag == 0; move to set 6, otherwise move to step 7.
6. Cat in seeking mode, start seeking mode process
   – Replicate the position of cats (suppose m) using SMP parameter, where, m=SMP.
   – similarity and dissimilarity between test-cases and grouped the test-cases into different clusters.
   – Evaluate the Fitness of catspositions (test-cases)
   – Compare the fitness ($Fit_S(X_S)$) of cats (test-cases)and the minimum one acted as best cat position (best test-case) and determine the other catspositions using best cat positionand store into $X_S$.
   – If$Fit_G(X_G) < Fit_S(X_S)$,
         $Fit_S(X_S) \leftarrow Fit_G(X)$ and $X_S \leftarrow X_G$
     Else
         $Fit_S(X_S) \leftarrow Fit_S(X_S)$and $X_{S,new} \leftarrow X_S$
     $Fit_G(X_G)$ is global best fitness and $Fit_S(X_S)$ is seeking best fitness.
7. Cat in tracing mode, starts tracing mode process
   – Cats velocity is computed through equation 1.
         $V_{new} = w * V_i(t) + r_1 * (X_S(t) - X_i(t)) + r_2 * (X_G(t) - X_i(t))$     (1)
     In equation 1, w is inertia weight, $V_i(t)$represents the i^th test case velocity, r describes through rand function (rand(0,1)), $X_G(t)$ represents best cat position (test-case) and $X_i(t)$is current cat position (test-case).
   – Update the position of catk using equation 2.
         $X_{new} = X_i(t) + V_{i,new}$     (2)
     In equation 2, $X_{i,new}$ represents new test case, $X_i(t)$ denotes i^th cat position (test case) and $V_{i,new}$ represents velocity of ith test case.
   – Determine the similarity and dissimilarity between test-cases and Grouped test-cases into different clusters through similarity and dissimilarity measure
   – Compute the fitness function ($Fit_T(X_T)$) and store the best positions of cats in a variable $X_T$.
8. If $Fit_T(X_T) < Fit_S(X_S)$
         $Fit_G(X) \leftarrow Fit_T(X_T)$and $X_G \leftarrow X_T$
   Else
         $Fit_G(X) \leftarrow Fit_S(X_S)$and $X_G \leftarrow X_S$
   ($X_G$)→ global best cat position, $Fit_T(X_T)$→ Tracing mode fitness, $Fit_S(X_S)$→ Seeking mode fitness and $Fit_G(X)$→ fitness of global best cat (test case).
9. Is termination condition met, stop and collect final solution, execute step 5.
```

Fit_S denotes fitness of seeking mode and Fit_T describes fitness of tracing mode. Fit_Gdenotes global best fitness of cat.
X_S describes best cats position in seeking mode and X_T describes best catsposition in tracing mode.$X_G$represents the global positions of cats.

3.4 Algorithm Explanation

CSO based prioritization algorithm starts with random initialization of cats population and main task of algorithm is to divide the test suite in optimal clusters, but having no prior information regarding clusters..The populations are selected randomly. In next step, the position and velocity vectors of CSO based prioritization algorithm can be defined. In this work, faults represented the cat position, while, execution can be used to describe the velocity of cat. The optimal solution for optimization problem can be either in term of minimization or maximization. This work considers the maximization as the solution for CSO based prioritization algorithm. Hence, the objective function can be described in terms of maximum faults detected with respect to minimum test cases. Further, an execution time is also adopted as performance measure to compute efficacy of algorithm and it should be kept minimum, So, along with maximum faults, keep in mind that execution time should be minimum. Further, a mutation operator is also used to generate the diverse position of cats throughout the execution of the program. The mutation operation changes the velocity and position of cats and provides more optimum results in terms of faults detection. The algorithm should stop its execution after reaching maximum iteration. The optimum solution in terms of maximum faults can be determined and also scheduling of test-cases is computed.

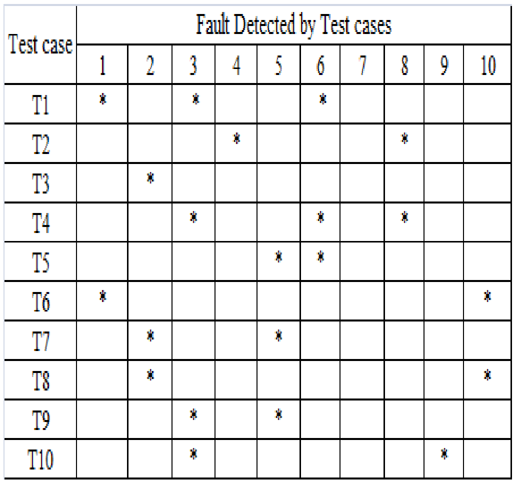
3.5 APFD Metric

The results of CSO based prioritization algorithm is computed through APFDmetric. This metric gives the results in terms of maximum faults determined through test-cases [18]. It can be ranges in between 0 to100. Considera test suite(T) with ntest-cases. F represents faultsetwhich can be boundedthrough test-cases. Suppose $T_i$denotes first test-case that can explore fault i.APFDcan be computed using following equation.

$$\text{APFD} = 1 - \frac{\text{T1} + \text{T2} + \text{T3} + \ldots + \text{Tm}}{\text{n}*\text{m}} + \frac{1}{2n} \qquad (3)$$

Table 1illustrates the ten test-cases. The test-cases are prioritized in order like$T_1 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2 \rightarrow T_5 \rightarrow T_8 \rightarrow T_9 \rightarrow T_7 \rightarrow T_6 \rightarrow T_{10}$;$T_3 \rightarrow T_1 \rightarrow T_2 \rightarrow T_6 \rightarrow T_5 \rightarrow T_4 \rightarrow T_8 \rightarrow T_{10} \rightarrow T_7 \rightarrow T_9$.The APFD equation is adopted for determining test-cases ordering. Furthermore, test-cases having higher APFD values can be selected from test suite. The faults detection can show the effectiveness of the test-cases. TCPcan be optimized through maximum faults with respect to minimum test-cases.

Table 1: Description of test-cases with seeded faults



Test case	Fault Detected by Test cases									
	1	2	3	4	5	6	7	8	9	10
T1	*		*			*				
T2				*				*		
T3		*								
T4			*			*		*		
T5					*	*				
T6	*									*
T7		*			*					
T8		*								*
T9			*		*					
T10			*						*	

Table 2: Execution time of test-cases

Test Case	Faults	Execution Time
T1	5	13
T2	6	11
T3	4	12.5
T4	3	10
T5	2	14
T6	3	9
T7	5	16
T8	4	8
T9	1	9
T10	2	11

4. SIMULATION RESULTS

This section illustratesperformance of CSO based prioritization algorithm for prioritization of test-cases using a given test suite. Test cases are prioritized through maximum faults with respect to minimum test-cases. Further, execution time is also considered one of important parameter to evaluate the effectiveness of CSO based prioritized algorithm. Hence, the execution time is associated with all test-cases. A total thirty-eight test-cases are designed in this work for a test suite(TS). Software reliability, software tools used, lines of codes, faults and efforts (in hours) can be described the attributes of test-cases. The proposed CSO based prioritized algorithm works in two steps. Initially, clustering task is performed on the test suite. The objective of clustering is to divide the test-cases into different cluster and each cluster contains the similar test-cases. So, the aim of clustering is to determine the subsets of similar test-cases, and in turn reduce the testing effort by considering either one or two test-cases from each cluster for testing purpose. Total 31 test-cases are defined in test suite and in the first step, these 31 test-cases are separated into five clusters. Further, it is stated that an automatic clustering procedure is adopted in this work and no prior information is required regarding number of clusters. The second step corresponds for the selection of test-cases for testing purpose. In this work, there test-cases are considered for detecting the faults in a specified program and test-cases are prioritized through faults detected. The test-case that determine the maximum faults can be assigned higher priority in the given cluster. Table 3 presents the faults detected through each selected test-cases from a given cluster and rank the test-case on the behalf of faults detected. This works considers two software application for evaluating the performance of CSO based prioritized algorithm. These application programs are written in object-oriented language and validated the proposed algorithm. Further, some faults are seeded into program to check the robustness and completeness of CSO based prioritized algorithm with other prioritization algorithms. The seeding faults in program is 10.Table 3 demonstrates the faults detected through each cluster an also presents the effectiveness of the test-cases for faults detection. Its revealed that cluster 2 detects more faults as compared to rest of clusters. Whereas, cluster 5 detects lower faults as compared to rest of clusters. Hence, the scheduling of test-cases can be given as 2→3→1→4→5.

Table 3: Faults selected through CSO for each test case

Cluster	Test Cases	No. of Faults
1	1	3
	2	5
	3	4
2	2	4
	5	6
	3	5
3	3	5
	4	6
	10	3
4	5	5
	6	3
	3	3
5	7	4
	8	2
	9	3

Table 4 depicts the results of CSO, K-means and agglomerative techniques. The test cases are divided into five numbers of clusters and eachcluster consists of three test cases. It is observed that CSO technique detects higher faults as compared to k-means and agglomerative techniques. It is also noticed that for few test cases, k-means algorithm detects higher faults than agglomerative techniques. On other hand , agglomerative techniques detects higher faults in compression to K-means such as for cluster 1 and test case 1, agglomerative technique detect two faults, but k-mean detects one faults and for same, CSO technique detects three faults. Table 5 demonstrates the comparison of execution time of all aforementioned techniques. It is revealed that CSO technique requires minimum time as compared to k-mean and agglomerative techniques to optimize the test cases. Moreover, agglomerative technique requires maximum time for optimizing test cases. Table 6 illustrates the success rate of each technique. It is seen that for most of test cases, CSO technique achieve hundred percent success rate. While, K-means achieves hundred percent success rates only for three test cases and agglomerative technique achieves hundred percent success rates for two test cases.

Table 4: Fault detection using CSO, K-Means and Agglomerative techniques.

Cluster	Test Cases	Fault Detected		
		CSO	K-Mean	Agglomerative
1	1	3	1	2
	2	5	3	4
	3	4	3	2
2	2	4	2	3
	5	6	4	3
	3	5	3	4
3	3	4	5	3
	4	5	3	6
	10	3	1	2
4	5	4	2	5
	6	2	3	1
	3	3	1	2
5	7	4	3	2
	8	2	2	1
	9	3	2	2

Table5: Comparison of execution time of CSO, K-Means and Agglomerative techniques.

Cluster	Test Cases	Execution Time		
		CSO	K-Mean	Agglomerative
1	1	10.4	11.6	12.1
	2	11.2	11.8	13.4
	3	12.33	11.2	14.2
2	2	15.75	16.8	17.1
	5	13.6	14.1	15.6
	3	10.2	11.3	10.6
3	3	10.2	9.4	10.5
	4	14.1	15.3	15.5
	10	11.2	13.4	14.1
4	5	9.2	9.4	10.2
	6	12.2	11.4	12.4
	3	10.3	10.6	11.15
5	7	18.55	20.3	20.5
	8	11.6	12.1	13.4
	9	10.2	10.5	11.4

Table 6: Comparison of success rate of techniques for each test case.

Cluster	**Test Cases**	**Success Rate**		
		CSO	**K-Mean**	**Agglomerative**
1	1	100%	33.3%	50%
	2	100%	60%	80%
	3	100%	75%	50%
2	2	100%	50%	75%
	5	100%	66.6%	50%
	3	100%	60%	80%
3	3	80%	100%	60%
	4	83.3%	50%	100%
	10	100%	33.3%	66.6%
4	5	80%	40%	100%
	6	66.6%	100%	33.3%
	3	100%	33.3%	66.6%
5	7	100%	75%	50%
	8	91.56 %	100%	50%
	9	100%	66.6%	66.6%

5. CONCLUSION

This work investigates the applicability of CSO based algorithm for prioritizing the test-cases. Furthermore, faults detection can be considered one of the performance parameters for evaluating the efficacy of CSO based prioritized algorithm. The working of prioritized algorithm is described using two steps. The first step corresponds to determine the different clusters of test-cases usingtest suite. The clustering of test-cases can be done through test-case attributes. The second step corresponds for evaluating the performance of test-cases. To achieve the same, three telecasts for each cluster are selected in random order. The efficacy of test-cases is evaluated using fault detection and execution time parameters. Simulation results showed that CSO based prioritized algorithm achieves at par results than other compared algorithms. Moreover, CSO algorithms also considers APFD metric for representing the fault detection. In future, TCP problem can be handled through more meta-heuristic algorithms.

References

1. Arafeen, M.J. and Do, H. **Test case prioritization using requirements-based clustering.** *Proceedings of the IEEE 6th International Conference on SoftwareTesting, Verification and Validation*, (ICST) Mar. 18-22, IEEE Xplore Press, Luembourg, 2013, pp: 312-321.
2. El-Koka, A., Cha K.H. and Kang,D.K. **Regularization parameter tuning optimization approach in logistic regression.***Proceedings of the 15th International Conference on Advanced Communication Technology (ICACT),*2013, pp: 13-18.
3. Tsai P. W, Chu, SC., and Jeng, S. P.**Cat swarm optimization,***In PRICAI,Trends in Artificial Intelligence*, Springer Berlin Heidelberg, 2006, pp. 854-858.
4. Tsai PW, Jeng-Shyang Pan, Shyi-Ming Chen, and Bin-Yih Liao. **Enhanced Parallel Cat Swarm Optimization Based on the Taguchi Method**, *Expert Systems with Applications*, Vol. 39, Issue 7, 2012 pp. 6309-6319.
5. Panda, G, Pradhan, P. M, and Majhi B. **IIR system identification using cat swarm optimization**, *Expert Systems with Applications*, Vol. 38, No. 10, 20111, pp. 12671-12683.
6. Pradhan, P. M, and Ganapati G.**Solving multi objective problems using cat swarm optimization'**, *Expert Systems with Applications*, Vol. 39, No. 3, 2012, pp. 2956-2964.
7. Santosa, B., and Ningrum, M. K. **Cat swarm optimization for clustering,***In IEEE International Conference of Soft Computing and Pattern Recognition* (SOCPAR'09), 2009, pp. 54-59.
8. Kumar, Y., and Sahoo, G.**An Improved Cat Swarm Optimization Algorithm for Clustering,** *In Computational Intelligence in Data Mining*, Vol. 1, 2015, pp 187-197.
9. Kumar, Yugal and Sahoo, G. (2015). **A Hybrid Data Clustering Approach based on improved Cat Swarm Optimization and K- Harmonic Mean Algorithm**, AI communications, Vol 28, No. 4, 2015, pp. 1-14.
10. Kumar, Y., & Sahoo, G. (2014). **A hybridize approach for data clustering based on cat swarm optimization**. *International Journal of Information and Communication Technology*, Vol. 9, No. 1, 2016, pp. 117-141
11. Harrold M, Gupta R, Soffa M.**A methodology for controlling the size of a test suite.***ACM Trans SoftwEngMethodo*, Vol. 2, No. 3, 1993, pp. 270–285.
12. Rothermel G, Harrold MJ.**Analyzing regression test selection techniques**. *IEEE Trans SoftwEng,*Vol. 22(8), 1996, pp. 529–551
13. Rothermel G, Untch RH, Chu C, Harrold MJ.**Test case prioritization: an empirical study.** In: *Proceedings of internationalconference of software maintenance*. 1999, pp 179–188
14. Wong WE, Horgan JR, London S, Agrawal H.**A study of effective regression testing in practice**. *Proceedings of 8th IEEE international symposium on software reliability engineering* (ISSRE' 97). 1997, pp 264–274, Albuquerque, NM.
15. Rothermel G, Untch RH, Chu C, Harrold MJ.**Prioritizing test cases for regression testing**. *IEEE Trans SoftwEng*, Vol. 27(10), 2001, pp. 929–948
16. Elbaum S, Malishevsky A, Rothermel G.**Prioritizing test cases for regression testing.** In: *Proceedings of international symposium on software testing and analysis*. 2000, pp 102–112.

17. Rothermel G, Untch RH, Chu C, Harrold MJ.**Prioritizing test cases for regression testing.***IEEE Trans SoftwEng,*Vol. 27(10), 2001, pp. 929–948.
18. Elbaum S, Malishevsky A, Rothermel G.**Test case prioritization: a family of empirical studies.** *IEEE Trans SoftwEng,*Vol. 28(2), 2002, pp. 159–182.
19. Elbaum S, Kallakuri P, Malishevsky A, Rothermel G, Kanduri S.**Understanding the effects of changes on the costeffectiveness of regression testing techniques.***J SoftwVerificationReliab,* Vol. 12(2), 2003, pp. 65–83.
20. Elbaum S, Rothermel G, Kanduri S, Malishevsky AG.**Selecting a cost-effective test case prioritization technique**.*Software Quality Journal*Vol. 12(3), 2004, pp. 185–210.
21. D. H, Rothermel G, Kinneer A.**Prioritizing Junit test cases: an empirical assessment and cost-benefits analysis**. *EmpirSoftwEng*Vol. 11, 2006, pp. 33–70.
22. Qu B, Nie C, Xu B, Zhang X.**Test case prioritization for black box testing. I**n: *The proceedings of 31st annual international computer software and applications conference*. IEEECS press,Beijing, 2007.
23. Park H, Ryu H, Baik J.**Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing**. In: *The proceedings 2nd international conference on secure system integration and reliability improvement*. IEEECS press, Washington, 2008, pp 39–46
24. Khan SR, Rehman I, Malik S.**The impact of test case reduction and prioritization on software testing effectiveness.***In: Proceeding of international conference on emerging technologies*.2009, pp 416–421.
25. Bajwa, J. K., & Kaur, R..**An Adaptive Approach For Test Case Prioritization In Regression Testing Using Improved Genetic Algorithm**, *An International Journal of Engineering Sciences*, 2017, pp. 1-17
26. Gladston, A., Nehemiah, K., Narayanasamy, P., & Kannan, A..**Test case prioritization for regression testing using immune operator.** *The International Arab Journal of Information Technology*, Vol. 13(6), 2016, pp. 1-7.
27. Tulasiraman, M., &Kalimuthu, V. (2018). **Cost Cognizant history based prioritization of test case for regression testing using immune algorithm**. *Journal of Intelligent Engineering Systems*, 11(1), 2018, pp. 221-228.
28. Maheswari, R. U., & Mala, D. J. **Combined genetic and simulated annealing approach for test case prioritization**. *Indian Journal of Science and Technology,*Vol. 8(35), 2015.
29. Bian, Y., Li, Z., Zhao, R., & Gong, D. (2017). **Epistasis based aco for regression test case prioritization.***IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 1(3), 2017, pp. 213-223.
30. Chen, J., Zhu, L., Chen, T. Y., Towey, D., Kuo, F. C., Huang, R., & Guo, Y.**Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering**. *Journal of Systems and Software,*Vol. 135, 2018, pp. 107-125.
31. Tahvili, S., Afzal, W., Saadatmand, M., Bohlin, M., Sundmark, D., & Larsson, S. **Towards earlier fault detection by value-driven prioritization of test cases using fuzzy TOPSIS.** *In Information Technology: New Generations* (pp. 745-759). Springer, Cham, 2016.
32. Hettiarachchi, C., Do, H., & Choi, B.**Risk-based test case prioritization using a fuzzy expert system.***Information and Software Technology*, Vol. 69, 2016, pp. 1-15.
33. Noguchi, T., Washizaki, H., Fukazawa, Y., Sato, A., & Ota, K.**History-based test case prioritization for black box testing using ant colony optimization**. *In 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST),*2015, pp. 1-2.
34. Jiang, B., & Chan, W. K.**Input-based adaptive randomized test case prioritization: A local beam search approach.***Journal of Systems and Software,*Vol. 105, 2015, pp. 91-106.
35. Konsaard, P., &Ramingwong, L.**Total coverage based regression test case prioritization using genetic algorithm.***In Electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON),* 2015, pp. 1-6.
36. Mei, L., Cai, Y., Jia, C., Jiang, B., Chan, W. K., Zhang, Z., &Tse, T. H. **A subsumption hierarchy of test case prioritization for composite services**. *IEEE Transactions on Services Computing*, Vol. 8(5), 2015, pp. 658-673.
37. Panichella, A., Oliveto, R., Di Penta, M., & De Lucia, A. **Improving multi-objective test case selection by injecting diversity in genetic algorithms.***IEEE Transactions on Software Engineering*, Vol. 41(4), 2015, pp. 358-383.
38. Panwar, D., Tomar, P., & Singh, V.**Hybridization of Cuckoo-ACO algorithm for test case prioritization.***Journal of Statistics and Management Systems*, 21(4), 2018, pp. 539-546.
39. Zhao, X., Wang, Z., Fan, X., & Wang, Z.A Clustering-**Bayesian network based approach for test caseprioritization.***In Computer Software and Applications Conference (COMPSAC),* 2015 pp. 542-547.
40. Tulasiraman, M., Vivekanandan, N., &Kalimuthu, V. **Multi-objective Test Case Prioritization Using Improved Pareto-Optimal Clonal Selection Algorithm.***3D Research*, Vol. 9(3), 2018, pp. 32.
41. Suri, B., & Singhal, S.**Understanding the effect of time-constraint bounded novel technique for regression test selection and prioritization**. *International Journal of System Assurance Engineering and Management*, Vol. 6(1), 2015, pp. 71-77.
42. S. Andrews. **An Investigation into Mutation Operators for Particle Swarm optimization**,*in Proc. Congr. Evol. Compt.*, 2015, pp. 1044 –1051.