Received: October 26, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Cost Cognizant History Based Prioritization of Test Case for Regression Testing Using Immune Algorithm

Megala Tulasiraman[1*] **Vivekanandan Kalimuthu[1]**

[1]*Department of Computer Science and Engineering, Pondicherry Engineering College, India*
* Corresponding author's Email: megelaganesh@pec.edu

Abstract: Regression testing is one among the strongest testing criteria which ensure the quality of the software under test. However, regression testing is too expensive due to execution of too many test cases .Test case prioritization is one of the traditional techniques which improve the regression testing by proposing a test case order that increases the rate of fault detection . In this paper, we propose a cost – cognizant history based test case prioritization approach that utilizes the historical information of the test cases like cost, fault identified by the test case and the severity of the identified fault for prioritization. Also an artificial immune system based, Clonal selection algorithm is proposed to find an effective test case order from existing test suite. To evaluate the proposed approach, controlled experiment were performed and the evaluation results indicates the test case order produced by proposed approach shows improvement in terms of average percentage of fault detected per cost.

Keywords: Regression testing, Clonal selection algorithm, Cost-cognizant, Prioritization.

1. Introduction

Software testing is the most significant and expensive activity throughout the software development life cycle model (SDLC). In modern software development, the software once developed has an extended life continuing to different versions. In each version new functionalities are added and changed based on the user's specification. As software is undergoing frequent changes the change induced in one functionality should not affect the other part of the code to ensure this regression test is performed. When an error occur in unmodified code due to the modification in one part of the code we call it regression error [1].The testing team performs regression testing to detect the regression error and to ensure the stability and quality of the software under test.

There are several techniques to perform regression testing one such technique is retest all. In retest all technique all the test cases in test suite are retested which is practically impossible due to time constraint. Other techniques include selection, minimization and test case prioritization. In test suite selection and minimization the test suite is reduced which results in decreased fault detection capability due to discarding of test cases. Whereas in prioritization the test cases are scheduled in an order such that higher priority test cases are executed earlier [2].Many existing technique have been proposed to prioritize test cases based on source code, the limitation of this method is, when the tester is not able to obtain the source code applying this method is difficult. Also some of the existing method considers uniform value for test case cost and fault severity [3] which is practically impossible. During each regression testing information about test cases are generated, these information are not effectively utilized by existing technique. In order to overcome the above stated limitations the cost –cognizant test case prioritization approach and history based prioritization techniques was proposed.

Existing prioritization techniques used different meta-heuristic algorithm like genetic algorithm, ant colony and particle swarm optimization for prioritizing test cases but none of the existing

technique uses artificial Immune system based algorithm for prioritization. Therefore in this paper we introduce Clonal selection algorithm (CSA) an artificial immune system based algorithm proposed by zuben and castro [4]. The CSA have been widely used in many applications such as pattern reorganization, intruder detection system and also in software testing activities such as test data generation and structural testing. Since CSA have been successfully applied in software testing activities we make an attempt to employ clonal selection algorithm in regression testing. Therefore in this paper we propose cost-cognizant history based prioritization of test cases using Clonal selection algorithm (CC-CSA) which utilizes historical information from the recent regression test to order test cases. The main contribution of this paper is as follow

(i)An immune system based prioritization technique to prioritize test cases is proposed.

(ii)Effectively utilizes the historical information of test cases created in latest regression testing.

The remaining section of the paper is structured as follow: Section 2 describes the background of Test case prioritization and gives a general overview of Clonal Selection Algorithm and reviews the related works in test case prioritization. Detailed working of proposed methodology is presented in Section 3. In section 4 the experimental results are analysed. And section 5 presents the conclusion and future work.

2. Background and related works

2.1 Test case prioritization

In test case prioritization approach the test cases are sorted in an order such that faults are detected earlier as possible. Initially the prioritization of test case problem was proposed by rothermel.et.al [1]
Given:

Ts, existing test suite selected for prioritization, PTs the set of all permutation of Ts, and f, an objective function from PTs to real numbers

Problem: find Ts' ϵ PTs such that

$$(\forall Ts'')(T''\epsilon PTs)(Ts''\neq Ts')[f(ts'\geq f(Ts')] \quad (1)$$

In test case prioritization, the important information of test case such as cost of executing test case and severity of fault are considered with a uniform value during prioritization which is considered as a shortcomings of existing technique. To overcome this issue a cost cognizant test case prioritization was proposed which takes different value for test case cost and severity of fault detected.

2.2 Clonal selection algorithm

Artificial immune system (AIS) is a new branch inspired by the immunological principle. The AIS aims to use the idea of immunology in various fields of science and engineering. Clonal selection algorithm is one of the AIS based algorithm which inspires and mimics the process of Clonal selection theory. The basic idea behind Clonal selection theory is to protect the host system from foreign particle such as viruses, bacteria and manmade molecules called as Antigen. The immune system produces antibody on identifying the antigen and proliferate to defend the antigen. Zuben and Castro [4] renamed Clonal selection theory to CLONALG and applied it in various engineering problems. In engineering problem the antigen represent element of a problem or problem to be solved. And the antibody represents the solution candidates. The two main features of Clonal selection algorithm (CSA) are hypermutation and cloning. CSA and genetic algorithm works in similar way but they differ in mechanism of generating new population. In CSA new population is generated using hypermutation operator as it does not support crossover operator whereas in genetic algorithm recombination operator such as crossover and mutation is applied to generate new population. The general algorithm for CSA is shown below.

General Algorithm for CSA

Begin

Step 1: Randomly initialize population with set of Antibodies A_0, A_1…..A_n

Step 2: Evaluate the affinity for each antibody A_0,.A_n

Step 3: Select antibody with higher affinity value from population

Step 4: Clone the antibody based on the affinity value

Step 5: Apply Hypermutation operator to create new population

Step 6: Clonal Selection: select superior antibody based on affinity value

Step 7: Termination: Repeat steps 3 to 6 until termination condition met

End

2.3 Related works

Test case prioritization technique schedules the test cases based on several objectives such as to improve the rate of fault detection. In recent years various prioritization techniques have been proposed and widely used by the software industries to improve their quality within the specified time[5].Most of the prioritization technique uses the information about source code to schedule the test case such total statement coverage, total functional coverage, modified statement coverage[6] and etc. The main limitation of these techniques are prioritization cannot be applied if the tester cannot obtain complete code. In order to overcome the limitation of code based prioritization cost cognizant test case prioritization and history base test case prioritization approaches were proposed. These prioritization technique schedules the test case based on software artefacts such as requirement covered, volality, cost and severity of fault.

Zhang and Nie et al. [7] designed a prioritization technique based on requirement priority and test case cost in which prediction of test case cost before execution is difficult. Kim et al [8] suggested a TCP technique, which orders test case based on historical value.however the technique is evaluated with couple of smaller programs on a function-based level concerning only a relatively low amount of tests .Park et al. [9] contributed a historical value based prioritization technique in which the test cases are prioritized based on the historical value generated. Yet the technique lacks in proving its efficiency by comparing only with functional coverage approach. Quet.al proposed prioritization of test case for black box testing using runtime information obtained. In recent study, a hybrid cost-cognizant history based test case prioritization technique [10] is designed which combines the cost-cognizant test case prioritization and history based test case prioritization. Very few prioritization technique were proposed in cost-cognizant history based test case prioritization. Yuchi et.al proposed a history based cost-cognizant prioritization of test case approach in which genetic algorithm is used to produce the scheduled order the main drawback of this technique is the number of iteration required to produce optimal result increases the chromosome size increase [11]. Recently many researchers have also effectively applied meta-heuristic algorithm such as genetic algorithm [12], ant Colony to search an optimal order from existing test case.Hence in this paper, a new methodology for test case prioritization with clonal selection algorithm is address above stated limitations.

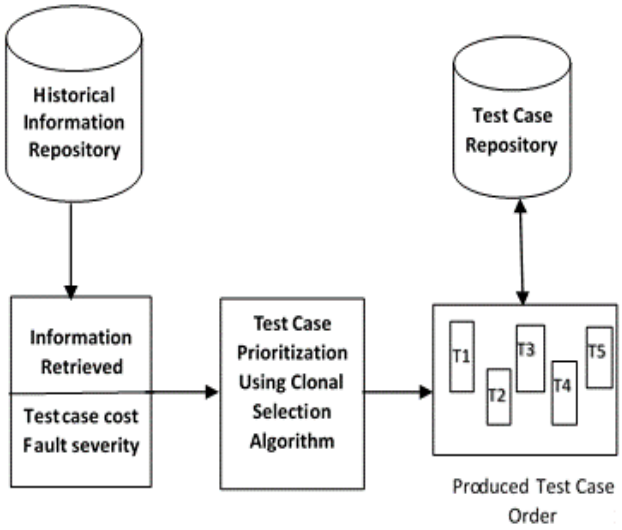

Figure.1 Overview of Cost-Cognizant history based Prioritization of Test Case approach using CSA

3. The proposed approach

In this section, the working of proposed approach is explained. The objective of the proposed approach is to find an effective test case order which is able to detect the severe fault earlier using the historical information of test cases. Since the proposed problem is a NP-hard problem it is solved using Clonal selection algorithm (CSA) a nature inspired meta-heuristic algorithm. The proposed approach works by taking the cost of executing test case and severity of detected fault as input. The historical information of each test case such cost of test case, fault identified by the test case and the severity of the detected fault are stored in historical information repository. In test case repository the generated optimal order for existing test suite is saved for reusing in future regression testing. The information about each test case is gathered from previous testing activities. Once prioritization is applied the proposed approaches inputs the information about test case to Clonal selection algorithm. The proposed CSA algorithm search for an effective test case order from the existing test suite using the information retrieved from the repository. Finally a scheduled test case order which is able to identify the severe fault earlier is generated as output. The overview of proposed approach is shown in Figure 1.

3.1 Proposed cost cognizant history based prioritization of test cases using clonal selection algorithm (CC-CSA)

The proposed CC-CSA algorithm schedules the test cases from the given test suite T, utilizing the historical information of test cases. In CSA the

antigen represents the objective of the problem and antibody represents the candidate solutions. Therefore in our proposed approach the antigens represent the best order to be searched out from existing test suite and antibody represent the each test case in the test suite. Algorithm for proposed CC-CSA approach is given below. The algorithm start by initializing the antibody Ab_i. Since random initialization is the most vital approach for initializing the population in proposed approach antibody is generated using random initialization.

Algorithm for CC-CSA

//After fetching the information test case cost *tc*, fault identified *fi*, and severity of identified fault *fsv* from historical information repository//

Input:

Test suite: *TS*

Size of the population: *PS*

Total Number of generation: *N*

Mutation Strategy: *MS*

Output:

Prioritized Order: Scheduled test order with highest affinity value in final generation

Begin

Step 1: Randomly initialize the population P_1

$P_1 \rightarrow$ generate population (*tc, fi ,fsv*)

Step 2: Determine the affinity value for each individual in population Pi and sort them from higher to lower

Affinity→ Evaluate affinity (*Pi,tc,fi,fsv*)

Step 3: Select antibody (test case) with highest fitness value

Step 4: Clone antibody (test case) with highest affinity value

Parent→ Parent $_c$

Step 5: Apply hypermutation operator (to create new population)

Parent $_c$→Mutation (Parent $_c$, *MS*)

Step 6: Repeat step 2 to step 5 until termination condition met

Return scheduled Test Order

End

3.1.1. Encoding

Antibodies can be represented using different encoding schemes for example decimal encoding, binary encoding and permutation encoding. In proposed approach permutation encoding is chosen for representing the antibody. In permutation encoding the antibody represents a sequence of test cases id which represents the execution order. For example figure.2 represents a sample antibodies were antibody A represents the execution order T6-T5-T4-T1-T3-T2 and antibody B represent an execution order T6-T4-T2-T1-T3-T5.

T6	T5	T4	T1	T3	T2	Antibody A

T6	T4	T2	T1	T3	T5	Antibody B

Figure.2 Representation of antibody

3.1.2. Affinity function

After initializing the population the affinity value of each test case is determined using the affinity function. The affinity of each antibody in the population is evaluated based on *tc* cost of test case, *fi* fault identified by the test case and *fsv* severity of fault as shown in step 2 of algorithm CC-CSA. Since the main objective of proposed approach is to schedule the test case in an order that identifies the severe fault earlier considering the cost of test case. The Average percentage of fault detected considering test case cost value for a test case is taken as the affinity value. If the program contains *f* faults and *n* test cases the affinity function is defined as follow.

$$fitness = \frac{\sum_{i=1}^{f}\left[fs_i \times \left(\sum_{j=TF_i}^{n} tc_j - (1/2)t_{TF_i}\right)\right]}{\sum_{j=1}^{n} tc_j \times \sum_{i=1}^{m} fs_i} \quad (2)$$

In Eq.(3) fs_i denotes fault severity of test case i,tc_j denotes the cost of the *jth* test case and Tf_i is the first case that detect the fault i . The test case which identifies the severe fault earlier will have the highest affinity value. Selection process is applied on initial population set to select set of antibodies from initial population for cloning process as shown in step 3 of CC-CSA algorithm. Roulette wheel selection is used for selecting the test cases therefore the test cases are sorted from highest value to lowest value in order to calculate the selection probability of each test cases. Selection probability for each antibody (test cases) is calculated as follow

$$S_k = \frac{fit_k}{\sum_{i=1}^{ps} fit_i} \quad (3)$$

S_k denotes the selection probability and fit_k represent the affinity value and *ps* denotes population size.

3.1.3. Cloning

After selecting the test cases, cloning process is applied as shown in step 4 of CC-CSA algorithm. Cloning is the process of replicating the antibodies with higher affinity value. The number of copies created for each antibody is determined using the Eq. (4).

$$Nc = \sum_{i=1}^{n} round(\frac{\beta.M}{i}) \quad (4)$$

In Eq.(4), the N_C represents the total clone generated for each antibody, β is a multiplying factor and M is total number of antibodies in population and function round(.) is used to round off the function value to nearest integer. For example consider M=50 and β=1, then the antibody with highest affinity will produce 50 clone and second highest with 25 and so on. The test cases are cloned based on the affinity value.

3.1.4. Hyper –mutation

In Clonal selection algorithm the next generation population is generated by hyper mutation process. The cloned antibodies are hyper mutated to produce offspring. In this context offspring represent different test case order. In hypermutation the copy of antibodies are mutated based on the affinity value. For antibody with higher affinity value, low mutation rate is applied and high mutation rate is applied for antibody with lower affinity value. In Single point mutation it is not possible to achieve hyper mutation therefore a mixed mutation strategy [13, 14] is applied. Figure 3 shows an example of hypermutation strategy.

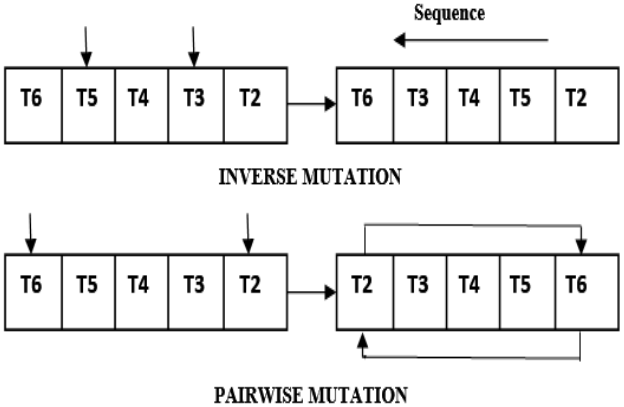

Figure.3 Example of mutation

Table 1. Characteristics of selected application

Version	LOC	No. of. Faults	No. of. Test Cases
1	11,450	8	300
2	10,278	7	300
3	9876	5	300
4	8770	7	300
5	6550	4	300

In mixed mutation offspring are generated based on the affinity value. For antibody with lower affinity value two stage mutation is applied and for antibody with higher affinity single stage mutation is applied. In two stage mutation, inverse mutation is applied followed by pairwise mutation therefore to creates offspring far away from parent. In Inverse mutation two antibodies are randomly selected and sub string within the points are reversed followed by pair wise mutation. In single stage mutation pair wise mutation is applied to produces offspring slightly different from parent antibody by changing the position of two selected antibody. The next generation is generated only through mutation operation. Step 2 to step 5 in proposed algorithm are followed until maximum generation is reached which produce an optimal test case order as output.

3.2 Historical information about test cases

In this section the historical information about test cases is discussed. The information include test case cost and severity of faults.

Test case cost: test case cost is the resource required for executing the test case. The resource is time required to execute the test case.

3.2.1. Severity of fault

In the proposed approach the severity of fault is assigned a value between scaling of 4 to 1.The highest severity value of 4 is given to most severe fault which lead to the failure of product and the product cannot be used until the fault is fixed. Severity value of 3 is given to moderate fault which does not affect the working of the product but should be fixed in later releases. The severity value of 3 is assigned to lower level fault which does not affect the working of product and the fault can be fixed in later version and severity value 1 is assigned to least level fault.

4. Experimental setup

To evaluate the performance of the proposed approach, a hospital management application developed by version square private.ltd is used. Hospital management provides function to perform

typical task such as billing, providing appointments, managing patient records and etc. The application is developed using Java programming language.

The characteristics of the application are mentioned in Table 1.A single test suite is used for testing all five sequential versions and the faults are seeded by hand. The cost of test case is considered as the execution time and the severity of each fault is created based on the functions criticality. The parameter used by the proposed approach is shown in Table 2.

In order to evaluate the proposed technique, four other techniques are chosen from existing test case prioritization technique for comparison and each technique is presented below.

a) Random prioritization: In random test case prioritization technique the test cases are randomly ordered [1].

b) Prioritization based on total functional coverage: In this technique the test case are ordered based on the functional coverage. Test case which covers the maximum number of functions is given the higher priority [11].

c) Prioritization using genetic algorithm: In GA based prioritization the test case are ordered by genetic algorithm based on the known information of the test cases [10]. The parameter used by this technique is shown in table 3.

d) Cost Cognizant prioritization of test cases based on Total functional coverage: Cost Cognizant test case prioritization based on functional coverage is an improvement over total functional coverage prioritization. In this technique each test case is awarded a value based on functional covered, severity of fault exposed and cost of executing the test case. The test case prioritized based on award value [13].

To validate the proposed approach average percentage of fault detected with respective to cost is taken as the evaluation metric. Since in our experiment, the proposed approach CC-CSA and the existing approaches such as random, optimal technique generates different test case order during each execution 500 prioritization were created for each method and the average of these prioritization is taken for experimental study.

Table 2. Parameter used by CC-CSA

Parameter	Value
Size of Population	200
Total number of generation	500
Mutation Operator	Inverse and Pairwise Mutation
Stopping Criteria	Maximum generation reached
Program coverage Weight	100
Test Adequacy criteria	Functional Coverage

Table 3. Parameter used by genetic algorithm

Parameter	Value
Size of Population	200
Total number of Generation	400
Crossover Rate	1
Mutation Rate	0.7
Program coverage Weight	100
Test Adequacy criteria	Functional Coverage

Table 4. APFDc value of prioritization techniques

Version	V1	V2	V3	V4	V5	Mean
Random	70.03	71.22	83	78.4	71.06	74.74
Tot-Fun	79.67	68.8	73.68	80.34	68.4	74.17
Optimal (GA)	96	90.39	95.2	97.06	96	94.93
CC-tot-fun	76.53	90.47	91.63	92.86	85.61	87.42
CC-CSA	88.9	94.24	96.76	97.03	98.87	95.16

5. Analysis of experimental result

In this section, the experimental result obtained during the evaluation of the proposed approach is presented. To validate the proposed approach average percentage of fault detected with respective to cost (APFDc) value of each technique is presented in Table 4.

As seen from table 4 the mean value of CC-CSA is higher than the random technique because of intelligent prioritization of CC_CSA. In random technique test cases are scheduled blindly whereas in CC-CSA the test case are scheduled based on previous history. Thus it shows 20.4% improvement over random technique and from figure 6 it is evident that the proposed approach outperforms than random ordering technique for all five version. The optimal technique also achieves better APFDc value than random technique. The random technique performs better in comparison with functional coverage technique and has 0.57% improvement but the difference is not significant.

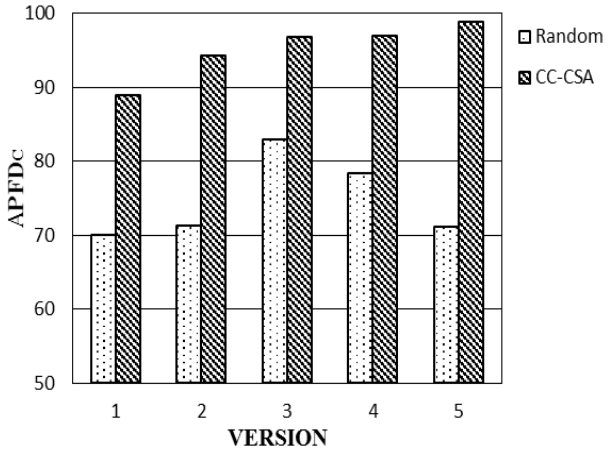

Figure.4 The Comparison of APFDc value of CC-CSA and Random

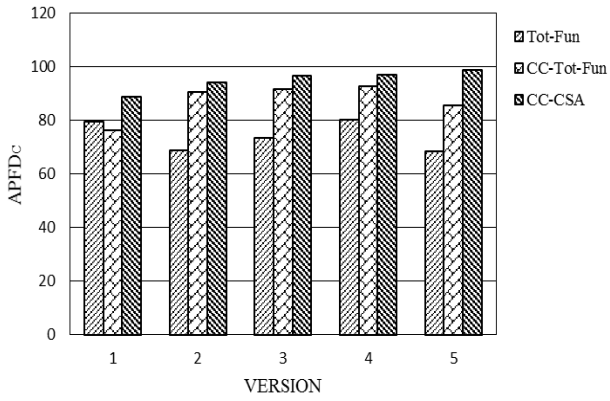

Figure.5 APFDc value of Functional coverage and CC-CSA

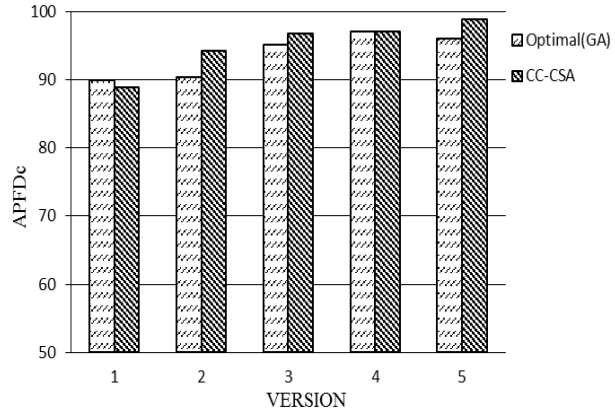

Figure.6 APFDc value of CC-CSA and Optimal (GA)

The total functional coverage method identifies maximum fault by scheduling the test case based on functional coverage criteria but fails to detect the most severe fault earlier. In contrast the CC-CSA order test case based on ability to find most severe fault. By comparing the proposed approach with total functional coverage the CC-CSA identifies severe fault earlier for all five version and shows

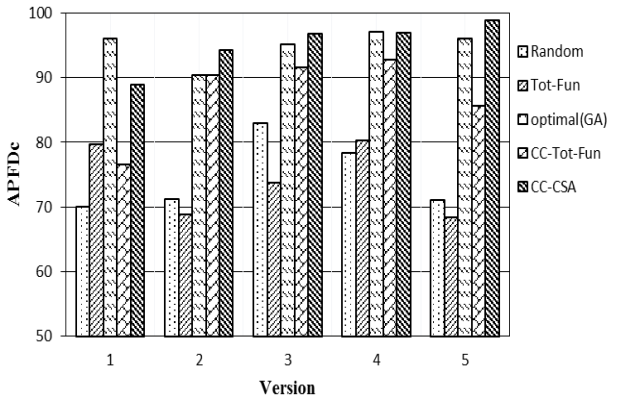

Figure.7 APFDc of all existing prioritization approach with CC-CSA

7.74 % improvement. Both the tot-fun and CC-tot-fun belongs to functional coverage based technique and the cc-tot-fun performs better than tot-fun for most version and is graphically represented in Fig. 7.

Both the Optimal and CC-CSA are meta-heuristic algorithm which finds an order by searching the existing test suite. Due to the high degree of parallelism and guided mutation of clonal selection algorithm it perform better than GA. From observing figure 8, the optimal (GA) technique has higher performance in version 1 and 3 and CC-CSA have outperformed better in version 2, 4 and 5.Considering table 4, the proposed CC-CSA approach has higher performance than all other existing approaches considered.

6. Conclusion

In this paper, we presented a Clonal selection algorithm based cost cognizant test case prioritization technique which uses the historical information of test cases for scheduling the test cases in an order that has great effectiveness in detecting severe faults early. The proposed system is evaluated using an industrial application with multiple versions. The result of this study demonstrated that newly proposed CC-CSA perform better than the random, optimal, and total-function and cost-cognizant total function approach in terms of APFDc. Even though Clonal selection algorithm is employed for first time in test case prioritization, from the experimental results it is evident that Clonal selection algorithm performs better by searching an order from existing test suite earlier than genetic algorithm for most cases. The two major contribution of this research is firstly, the historical information of the test cases is effectively used for prioritization. Secondly, Clonal selection algorithm is introduced in prioritization technique. With proposed approach, software industry can manage the regression testing activities and

prioritize test case without the source code. Further in this work, the proposed approach effectiveness can be improved by fine tuning the parameters of Clonal selection algorithm. Therefore are future study will be considering different mutation and cloning strategies to increase the effectiveness of proposed approach.

References

[1] G. Rothermel, U.C. Chu, and M. Harrold, "Test case prioritization", *IEEE Transactions on Software Engineering*, Vol. 27, No. 10, pp. 929–948, 2001.

[2] H. Srikanth, "Requirement based test case prioritization", In: *Proc. of Student Research Forum in 12th ACM SIGSOFT Internationa1 Symposium on the Foundations of Software Engineering*, Newport Beach, California, 2004.

[3] S. Elbaum, G. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization", In*: Proc. of the 23rd International Conf. On Software Engineering*, Ontario, Canada, pp. 329–338, 2001

[4] L. N. De Castro and F. J. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 3, pp. 239-251, 2002.

[5] S. Elbaum, G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies", *IEEE Transaction on Software Engineering*, Vol.28, No.2, pp.159-182, 2002.

[6] X. Zhang, C. Nie, B. Xu, and B. Qu, "Test case prioritization based on varying testing requirement priorities and test case costs", In: *Proc. of the 7thInternational Conf. on Quality Softwa*re (QSIC'07), pp.15–24, 2007.

[7] J.M. Kim and A. Porter, "A history-based test Prioritization technique for regression testing in resource constrained environments", In: *Proc. of the 24th International Conf. on Software Engineering*, pp.119-129, 2002.

[8] H. Park, H. Ryu, and J. Baik, "Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing", In: *Proc. of the 2nd International Conf. on Secure System Integration and Reliability Improvement*, Yokohama, Japan, pp. 39–46, 2008

[9] C.T. Lin, C.D. Chen, C.S. Tsai, and G.M. Kapfhammer, "History-based test case prioritization with software version awareness", In: *Proc. of the 18th International Conference on Engineering of Complex Computer Systems*, pp.171-172, 2013.

[10] Y.C. Huang, K.L. Peng, and C.Y. Huang, "A history-based cost-cognizant test case prioritization technique in regression testing", *Journal of Systems and Software*, Vol. 85, No. 3, pp.626–637, 2012.

[11] S. Elbaum, A. Malishevsky, and G. Rothermel, "Prioritizing test cases for regression testing", In: *Proc. of the ACM International Symposium on Software Testing and Analysis*, pp. 102–112, 2000.

[12] R. Krishnamoorthi, and S.A.S.A. Mary, "Factor oriented requirement coverage based system test case prioritization of new and regression test cases", *International journal of Software Technology*, Vol.51, No.4, pp. 799–808, 2009.

[13] G. Malishevsky, J.R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant Test Case Prioritization", Technical Report TR-UNL-CSE-2006-0004 University of Nebraska-Lincoln, 2006.

[14] B.H. Ulutas and S. Kulturel-Konak, "Assessing hypermutation operators of a clonal selection algorithm for the unequal area facility layout problem", *Engineering Optimization*, Vol. 45, No. 3, pp.375–395, 2013.

[15] T. Megala and K. Vivekanandan, "Clonal Selection Algorithm with Adaptive Lévy Mutation Operator", In: ICIA-16 *Proc. of ACM International Conf. on Informatics and Analytics*, pp.43-50, 2016.

[16] K.R. Walcott, M.L. Soffa, G.M. Kapfhammer, and R.S. Roos, "Time-aware test suite prioritization", *In: Proc. of the ACM International Symposium on Software Testing and Analysis*, pp. 1–12, 2006.

[17] M.J. Arafeen, "Test case prioritization using requirements based clustering", In: *Proc. of the 6th IEEE International Conference on Software Testing, Verification and Validation*, pp. 312–321, 2013.

[18] Z. Sultan, R. Abbas, S. Nazir, and S.Asim, "Analytical Review on Test Cases Prioritization Techniques: An Empirical Study", *International Journal of Advanced Computer Science and Applications*, Vol.8, No.2, pp. 293-302, 2017.

[19] V. Cutello, G. Nicosia, and M. Povene, "Real coded clonal selection algorithm for unconstrained global optimization using a hybrid inversely proportional hypermutation operator", In: *Proc. of the ACM on applied computing*, France, pp.950–954, 2006.