Journal of Statistics & Management Systems
ISSN 0972-0510 (Print), ISSN 2169-0014 (Online)
Vol. 26 (2023), No. 1, pp. 159–167
DOI : 10.47974/JSMS-955

A novel real coded GA for regression testing

Deepti Bala Mishra *
Dharashree Rath †
Swarnalipsa Parida §
Gandhi Institute of Technological Advancement Autonomous College
Bhubaneswar 752054
Odisha
India

Abstract

Regression testing is an activity performed in maintenance level of software. Growing of software test suite results in high cost and which also damages the software system. Test case prioritization is used to prioritize the most appropriate test cases. This research paper presents a test case prioritization and minimization technique by taking various factors involved such as risk exposure, requirement factor, statement coverage data and execution time. There are different factors of test cases such as statement coverage data, requirements factors, risk exposure and execution time. Genetic algorithm is used to prioritize test cases and further, test cases are minimized based on some given time constraint. The effectiveness of the prioritized test cases is measured by APSC metric.

Subject Classification: *68T07.*

Keywords: *Regression testing, Test case prioritization, Risk exposure, Genetic algorithm, Requirement factor, Statement coverage.*

1. Introduction

Software testing requires most significant time and effort in the Software Development Life Cycle (SDLC). Regression testing is the maintenance level of software testing. It is tested to check whether the changes made in an existing software should not play any uncoverable impact on quality of the software. Generally, when the software product

*E-mail: dbm2980@gmail.com (Corresponding Author)
† E-mail: dharashree96@gmail.com
§ E-mail: pswarnalipsa@gmail.com

is executed by the customer and they noticed some disturbances in that product at the time of run i.e., run-time error, in that situation the product is again handed over to the software industry for modification. When changes made in existing software the regression testing concept arises [1,2]. Basically, when the modification is made in any software during regression testing, the tester should make sure that after the modification there should not be any unfavorable impact on quality of the software [3, 4]. Regression Testing technique, which is one of the important testing techniques is performed on the updated version of software.

In a software product, as the test suites increases, it leads to increase in the test cases also, which results in time consuming and costly development. So, to avoid these types of circumstances, the developers go through some techniques which can be performed during regression testing like; test case selection, test case prioritization and test case minimization [5].

Generally, in regression testing technique the program under test is divided into some sub-programs or suites, which are known as test-suites. These test suites are ready to test with the help of existing test cases and some newly generated test cases that are developed by the tester during testing the updated version [6, 7]. The main aim of the regression testing is that the modified software does not affect the execution of the previous software though changes take place in the coding part. An approach being used named as "retest-all approach", which helps to execute the test cases in the test-suite. As the growing of test cases is directly proportional to the growing of test suites, it results in a costly process and may affect the whole cost of the software product development [8, 9].

The rest part of this paper is arranged as: a short description on Genetic Algorithm (GA) drawn in Section 2. Section 3 describes the literature review of some existing works on regression testing techniques and in Section 4, the proposed algorithm is defined. The experimental setup along with the result analysis are discussed in Section 5. Finally, the conclusion and some future scopes are written in Section 6.

2. Genetic Algorithm (GA)

GA was proposed by professor J. Holland in year 1975. The main goal of GA is to work on the structure of objects. Using gene reproduction, cross-over, mutation gives effective enhancement to the environment. First the population of individual is initialized [10]. Mutation can be introduced with some probability for each individual. Then the fitness function of each individual is evaluated [11].

This research paper presents, permutation encoding [12, 13], average crossover, and insertion mutation operators of GA. The permutation encoding is used to create Chromosomes for the proposed method, average Crossover and Insertion Mutation operators [11] are used to find the new offspring chromosome.

3. Related Work

Till date, so many works have been referred and studied on test case prioritization and minimization in regression testing. A few of the relevant research is presented in this section.

Deepti et. al. [12] have proposed an approach for test case prioritization using Genetic Algorithm (GA). Authors have also presented a novel code-based prioritization technique, which minimized the test cases with maximum mutant coverage. The proposed technique takes the prioritization factors like statement coverage, fault exposing potential award and mutant coverage. The experiment has been done on a small case study as the Triangle Classifier Problem (TCP) and GA has been used for test case prioritization and minimization purpose. *Chi et. al. [6]* have developed a new approach Additional Greedy Method Call sequence (AGC) to guide test case prioritization effectively. In the proposed method, dynamic function call states of the art TCP technique in different aspects. Their reported results show that the proposed AGC out performs in scalability factors than other existing methods. The bug detection capabilities are also high and they have achieved a highest mean APFD value. *Soumen et. al. [4]* proposed a honey bee swam intelligence algorithm for test case optimization in regression testing. The proposed algorithm has been designed to enhance the fault detection rate in minimum time. The proposed nature inspired algorithm was implemented on two VB projects with incorporating some faults into them. Authors have discussed various strategies of prioritization in regression testing like no order, random order, reverse order etc. The proposed method yielded highest APFD value in comparison to other prioritization techniques. Wang et. al. [1] have proposed a technique by using an information flow that transmits along with some different components of a software i.e., Risk base Test case Prioritization (RI-TCP). Then according to the dependencies, the proposed algorithm was mapped into Class Level Directed Network Model based on Information Flow (CDNMIF). After that each test case priority was calculated by the help of total sum of the risk index and they reached on a conclusion that the RI-TCP technique gives a large rate of fault detection

capacity than other. *Sharma et. al. [5]* proposed an algorithm for test case prioritization by comparing different existing greedy approaches, and not got satisfactory results. Further, a hybrid algorithm, GA with hill climbing method has been proposed to find the optimal results. Different coverage factors and FEP value of test cases are taken for prioritization. The APFD metric has been used to find the performance of the proposed hybrid method and the reported results found better in terms of highest fault detection rate.

4. Proposed Algorithm

In this section, a GA based algorithm has been developed for test case prioritization. A multi objective GA is also used to minimize and prioritize the test cases, which is based on the fitness value and total execution time. Some test case-oriented factors like requirement value, statement coverage is taken for prioritization and the execution time of each test case is taken for minimization purpose.

The proposed algorithm initialize population as test case sequence and calculate the fitness function of each chromosome. Selected two best populations based on fitness Then average crossover and insertion mutation operators are applied based on some probability factor. The duplicate test cases are removed through insertion mutation operator. Finally, the prioritized and minimized test cases are generated as the optimal solution.

5. Experimental Setup

For the experiment, different GA parameters are taken which are described in Table 1 and the coding is done in Dev C++ IDE-5.11. The population are selected on the basis of fitness function. The permutation encoding is used for creating initial chromosomes. The fitness function is defined as the summation of all test case factors, shown in Equation (1) and Equation (2). Higher fitness values are taken as best population and are selected for next generation.

$$\text{Maximize} \qquad F(x) = \sum_{i=1}^{n} \frac{TCW_i}{TCO_i} \tag{1}$$

$$\text{Subject to}: T_1 + T_2 + T_3 + \ldots\ldots T_n \leq \text{Total Execution Time} \tag{2}$$

Table 1
GA Operators

Sl No.	Operators	Value
1	Input	No. of Test Cases
2	Total Population	10
3	Coding	Permutation
4	Crossover and rate	Average Crossover (AX), 0.8
5	Mutation and Rate	Insertion Mutation, 0.02
6	No. of Generation	5

$$TCW = \text{Total Statement Covered} + \text{Priority Factor Value} + \text{Risk Exposure} \quad (3)$$

TCW can be evaluated by putting Requirement Priority Factor, Statement Coverage Information and Risk Exposure Value.

Were, TCOi= test case order of i^{th} test case, and n = total number of test cases

5.1 *Banking Management System*

The Banking Management System is a software that performs different operations in a bank such as New_Entry( ), Deposit( ), Withdraw( ) and Display_Statement( ). The different test cases generated are listed in

Table 2
Test cases generated for Banking Management System

Test case ID	Description
T1	Invalid Entry
T2	Valid Entry
T3	Correct Display
T4	Incorrect Display
T5	Unsuccessful Deposit
T6	Successful Deposit
T7	Invalid Account for withdraw
T8	Valid Account but insufficient balance for withdraw
T9	Withdraw successful
T10	Exit (No operation is done)
T11	Invalid Operation

Table 2 and the priority factor values for each requirement are shown in Table 3. In this project all the sequential statements are combined to form a statement unit. The statement unit coverage and execution time of each test case are listed in Table 4.

Table 3

Priority Factor Value for Requirements

Requirements	Total Priority Factor Values	Total Risk Exposure Value
R1	26	323
R2	14	162
R3	28	94
R4	23	44
R5	21	77
R6	16	223
R7	29	333
R8	27	306
R9	30	342
R10	7	41
R11	9	13

Table 4

Statement Unit Coverage and Execution Time of Different Test Cases

Test case ID → Statement Unit No. ↓	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
U1	#	#	#	#	#	#	#	#	#	#	#
U2	#	#	#	#	#	#	#	#	#	#	#
U3	#	#	#	#	#	#	#	#	#	#	#
U4	#	#	#	#	#	#	#	#	#		
U5	#	#									
U6	#										
U7		#									
U8	#	#									
U9			#	#							
U10				#							
U11			#								

Contd...

U12			#	#							
U13					#	#					
U14					#						
U15						#					
U16					#	#					
U17							#	#	#		
U18							#				
U19								#			
U20									#		
U21								#			
U22								#	#		
U23										#	
U24											#
Execution Time (In ms)	81	79	68	59	65	72	60	61	67	26	24

5.2 *Result Analysis*

For the project the prioritized order of test cases is T9, T1, T8, T7, T2, T6, T3, T5, T4, T10, T11 and the minimized test cases and their coverage information are shown in Table 5.

Table 5
Optimized Test Cases and their Coverage

Test case Factors (Coverage)	**Test cases**							**% Covered**
	T1	T9	T8	T7	T2	T6	T3	
Statement Unit Covered	7	6	7	6	6	6	6	83.33%
Requirement Covered	26	30	27	29	14	16	28	74%
Risk Covered	323	342	306	333	162	223	94	93.06%
Total Execution Time (in ms)	81	67	61	60	79	72	68	<=490 ms

Effectiveness of proposed Method
APS_UC(Non-Prioritized) =1 – {(1+1+1+1+1+1+2+1+3+4+3+3+5+5+6 +5+7+7+8+9+8+7+10+11 / (24*11)} + {1/ (2*11)}
= 1 – {110 / 284} + {1 / 22}
= 1 – 0.387 + 0.045
= **0.656**

APS_UC(Prioritized) = 1- {(1+1+1+1+2+2+5+2+7+9+7+7+6+8+6+6+1+4+3+1+3+1+10+11 / (24*11)} + {1/ (2*11)}
= 1 – {105 / 284} + {1 / 22}
= 1 – 0.369 + 0.045
= **0.676**

The APS_UC measures of prioritized order of test cases is higher than the non-prioritized order. The performance result shows that, the APSC values of prioritized test cases is higher that the non-prioritized test cases for the project Banking Management System.

6. Conclusion

In this paper a new approach is used for test case prioritization by considering total statement coverage, requirement priority factor value, risk exposure and execution time. GA is used to prioritize and minimize test cases according to the given time constraint. The proposed method is implemented on the Java based Projects of LOC-185. The efficiency of the proposed method is measured by the performance metric APSC.

In future, the proposed approach will implement on object-oriented software for prioritization and minimization. The effectiveness of the proposed method can be measured by using APFD and $APFD_C$ metrics for early detection of severe faults, by incorporating the cost factor.

References

[1] Wang, Y., Zhu, Z., Yang, B., Guo, F., & Yu, H. (2018). Using reliability risk analysis to prioritize test cases. *Journal of Systems and Software,* 139, 14-31.

[2] Panwar, D., Tomar, P. and Singh, V., 2018. Hybridization of Cuckoo-ACO algorithm for test case prioritization. *Journal of Statistics and Management Systems,* 21(4), pp.539-546.

[3] Jain, N., Porwal, R., Kumar, S., Varshney, S. and Saraswat, M., 2019. Automatic data flow class testing based on 2-step heterogeneous process using evolutionary algorithms. *Journal of Statistics and Management Systems,* 22(7), pp.1315-1348.

[4] Nayak, S., Kumar, C., Tripathi, S., Mohanty, N. and Baral, V., 2021. Regression test optimization and prioritization using Honey Bee optimization algorithm with fuzzy rule base. *Soft Computing*, 25(15), pp.9925-9942.

[5] Sharma, N. and Purohit, G.N., 2014, December. Test case prioritization techniques "an empirical study". In 2014 International Conference on High Performance Computing and Applications (ICHPCA) (pp. 1-6). IEEE.

[6] Chi, J., Qu, Y., Zheng, Q., Yang, Z., Jin, W., Cui, D. and Liu, T., 2020. Relation-based test case prioritization for regression testing. *Journal of Systems and Software,* 163, p.110539.

[7] Khatibsyarbini, M., Isa, M. A., Jawawi, D. N., & Tumeng, R. (2017). Test case prioritization approaches in regression testing: A systematic literature review. Information and Software Technology.

[8] Aggarwal, K.K., Singh, Y. and Kaur, A., 2005. A multiple parameter test case prioritization model. *Journal of Statistics and Management Systems,* 8(2), pp.369-386.

[9] Mani, P. and Prasanna, M., 2017. Validation of automated test cases with specification path. *Journal of Statistics and Management Systems,* 20(4), pp.535-542.

[10] Ray, M., & Mohapatra, D. P. (2014). Multi-objective test prioritization via a genetic algorithm. *Innovations in Systems and Software Engineering,* 10(4), 261-270.

[11] Mishra, D.B., Mishra, R., Acharya, A.A. and Das, K.N., 2019. Test case optimization and prioritization based on multi-objective genetic algorithm. In Harmony Search and Nature Inspired Optimization Algorithms (pp. 371-381). Springer, Singapore

[12] Mishra, D.B., Panda, N., Mishra, R. and Acharya, A.A., 2019. Total fault exposing potential based test case prioritization using genetic algorithm. *International Journal of Information Technology*, 11(4), pp.633-637.

[13] Yue, D., Zhang, H. and Dou, C., 2021. Cooperative Optimal Control of Hybrid Energy Systems. Springer Verlag, Singapore.