INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Performance Evaluation of "Ga-Fc" Technique for Aspect-Oriented Software System

Susheela Hooda[1]*, **Vikas Lamba**[1], **Amandeep Kaur**[1]

1 Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India

OPEN ACCESS

Received: 21.06.2021

Accepted: 08.11.2021

Published: 04.12.2021

Citation: Hooda S, Lamba V, Kaur A (2021) Performance Evaluation of "Ga-Fc" Technique for Aspect-Oriented Software System. Indian Journal of Science and Technology 14(41): 3074-3081. https://doi.org/10.17485/IJST/v14i41.1141

* **Corresponding author**.

susheela.hooda@chitkara.edu.in

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives : Exhaustive testing requires more effort and consumes lots of time of the software testers. Recently, "GA-FC" technique has been proposed to generate test cases optimally for Aspect-Oriented Software System (AOSS). This paper evaluates the performance of the proposed "GA-FC" technique. **Method**: To analyze the performance of "GA-FC" technique, two parameters namely, Effectiveness of Test Suite Minimization (ETSM) and Aspectual Branch Coverage (ABC) have been used against GA and FC technique individually. Findings: - GA-FC technique has been applied on three case studies and the obtained results reveal that GA-FC technique reduces the testing efforts by producing the minimum number of test cases and time of the tester. Novelty: "GA-FC" technique generates the minimum number of test cases by composition of metaheuristic techniques.

Keywords: AspectOriented Software System; Aspectual Branch Coverage; Genetic Algorithm; Fuzzy Clustering Algorithm; AspectOriented Software Testing

1 Introduction

Verification and validation activities are performed throughout the life cycle of the software development process. These activities are generally used to evaluate the overall correctness of the product [(1)]. Therefore, both activities improve the reliability of the software and make ensure that the developed system fulfills the software requirement specification and satisfy the customer's needs [(2)]. Only good software testing process improves the quality and reliability of the software. However, exhaustive testing is impractical and not feasible, as even small programs may have a number of test cases that can be extremely large [(3)]. Therefore, exhaustive testing is not possible due to time and cost constraint. Therefore, the need of the hour is to develop the most efficient and effective technique which can reduce the time, cost and effort. Consequently, selection of the best optimal test data to the software is the main challenge against the software testers.

Now a days, Artificial intelligent techniques have been used in every walk of life such as intelligent used in transportation system, software testing, cloud computing, healthcare, agriculture [(4–6)]. Reduction of the size of the test suite is known as test suite

minimization technique. In test suite minimization approach, test suite is selected from the original large test suite which contains a smaller number of test cases. Thereafter, this minimized test suite is used to test the software effectively and efficiently on the basis of certain criteria. Thus, minimized test suite reduces the redundant test cases or eliminates the test cases that are of no use from the original test suite. In this way, test suite minimization technique directs to reduce the testing cost [7].

However, UML (Unified Modeling Language) is gaining more attention of the researchers in the field of software engineering. UML provides a blueprint of the system that can be easily understood by the non-technical persons too. Consequently, UML activity diagram is used to depict the workflow of the system. Independent paths can be derived from UML activity diagram. These independent paths can be used as test cases but generated test cases can be large in size. So, one need to be reduced the number of test cases on the basis of the specified criteria using some systematic procedure. According to Harrold et.al. [7], definition of test suite minimization can be defined as: "Let T be a test suite with a set of test cases $\{t_1, t_2, —-, t_n\}$, and a set of test requirements R= $\{r_1 Ur_2 U———Ur_n\}$.Therefore, T1 is a subset of T satisfying a requirement criteria R1 where R1= {r11Ur21U——-Ur1n1} Many researchers have tried to optimize the test case selection process using the different techniques [8–10]. But, still it is very challenging because current solutions are not sufficient as they are limited in scope. A number of evolutionary and non-evolutionary algorithms have been developed to minimize the size of the test suite for a given problem in a traditional programming. Researchers have put their effort to optimize the large test suite by using evolutionary and non-evolutionary algorithms [11–15] in traditional programming. But literature of aspect-oriented software testing techniques indicates that a very few works has been done towards the test suite minimization using the evolutionary or non-evolutionary algorithms and moreover, very few frameworks have been developed to test the validation of aspect-oriented software [16–25]. Recently, a novel technique "GA-FC" has been proposed to reduce the number of test cases from the large number of test cases by using the characteristics of two metaheuristic algorithms; GA and FC [26]. This approach takes the advantage of both algorithms to make the testing process efficient. Optimality and fast processing over large solution space property feature of GA and soft clustering feature of FC to get more significant results are taken into consideration. This paper presents a performance evaluation of the proposed "GA-FC" techniques on three well known problems of aspect-oriented software system namely, a) Automatic teller machine, 2) Online Banking System and 3) On line Shopping for minimization of test cases on the two parameters such as 1) ETSM and 2) ABC.

2 Proposed Model for "GA-FC"

In this section, an aspectual UML activity diagram-based test suite optimization model for proposed "GA-FC" technique for AOP has been developed. The proposed model for "GA-FC" technique has been revealed with an aspectual UML activity diagram and an evolutionary computation technique namely, Genetic Algorithm and Fuzzy Clustering. Test suite optimization techniques will not always prove their success to find the best results due to limited resources. Hence, it can be assumed that integration of automated model-based test case generation and selection with optimization technique may produce one or more effective test cases. Therefore, in this work, an effort has been taken to develop test suite optimization technique using aspectual UML activity diagram for AOP. Results obtained from the proposed "GA-FC" technique has proven success in the field of Aspect-Oriented Software testing.

Figure 1 depicts the detailed sketch of the model for "GA-FC" technique, in which test cases are generated from the requirement model of the case study and evaluated with the test adequacy criteria i.e. Aspectual Branch Coverage (ABC) Criteria. If generated test case is competent based on test adequacy criteria then passes to the next phase otherwise more iterations are needed to generate adequate test case. The following steps have been followed in the proposed "GA-FC" model:-

- Construct an aspect weaving or aspectual UML activity diagram from the source code of an AOP and transform into CFG. Definition of CFG is stored in adjancey matrix.
- Generate effective and efficient test cases from the aspectual UML activity diagram by using the application of GA. An objective (i.e. Prioritized Fitness Value (PFV) function is defined as [18]:

$$PFV_i = IF_i + W_i$$

where IF_i=FAN_{in}*FAN_{out} of each node and W_i=Weight of each node according to node type.

- To obtain the more specific or accurate results, the output of GA has been used as an input for FC algorithm in the proposed "GA-FC" technique and performed evolutionary computation to achieve more optimized test suite [26].

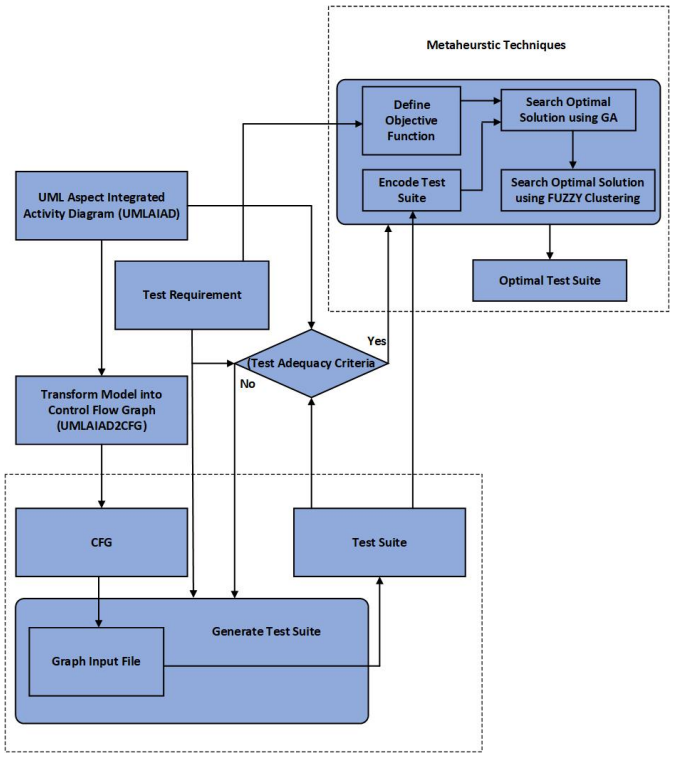

Fig 1. Detailed Sketch of the Model for Proposed "GA-FC" Technique

3 Performance Evaluation of "GA-FC" Technique

The performance of the proposed "GA-FC" test suite minimization technique has been evaluated against GA (Genetic Algorithm) and FC (Fuzzy Clustering) techniques individually for test suite minimization. GA and FC have been applied in software testing to identify reduced, prioritized and optimized test sets for a given problem. The main purpose is to perform a rigorous and comprehensive comparative study of GA and FC and "GA-FC". To assess the comparative performance of the proposed "GA-FC" approach, three well-known AOP case studies are used for experimental evaluation of the proposed "GA-FC" technique. To conduct the comparative evaluation of proposed "GA-FC" technique, the following two parameters have been taken:

1. Effectiveness of Test Suite Minimization (ETSM)
2. Aspectual Branch Coverage (ABC) criteria

ETSM basically deals to identify and eliminate the redundant test cases from the test suite [11]. Notion for ETSM:

$$ETSM = 1 - \frac{Numberoftestingpaths \in aminimizedtestsuite}{Totalnumberoftestingpaths \in originaltestsuite} * 100$$

The impact of test suite minimization can be measure using ABC parameter which "Measure as the percentage of covered aspectual branches among all branches" [13]. This parameter is one of the most popular which has been used commonly in existing testing techniques for AOSS [15].

Notion for ABC:

$$ABC = \frac{Numberoftestcasescenarioswhichcoveredaspectualbranches}{TotalNumberofTestCaseScenarios \in aTestSuite} * 100 \quad (3)$$

Three well-known case studies of AOP namely "Case Study-1: Automatic Teller Machine [9,26]", "Case Study-2: Online Banking System [25]" and "Case Study-3: On line Shopping System [24] have been taken to assess the experimental analysis of the proposed "GA-FC" technique against GA and FC.

3.1 Case Study-1 "Automatic Teller Machine"

The proposed "GA-FC" technique was applied to case study-1, "Automatic Teller Machine[13]". Case study-1 has one aspect node: "Authorization". From the UML activity diagram of "Automatic Teller Machine" generated seven independent testing paths[26]. Table 1 depicts the five independent testing paths and their PFVs are correspondingly for case study-1.

Table 1. Independent Testing Paths with their PFVs for case study-1

Test Case Id	Testing Paths	$PFV_i=W_i+IF_i$
T1	" 1->2->3->4->5->15->16->17->18"	25
T2	" 1->2->3->4->5->6->7->8->13->14->15->16->17->18"	39
T3	"1->2->3->4->5->6->7->8->9->10->12->13->14->15->16->17->18"	49
T4	"1->2->3->4->5->6->7->8->9->11->12->13->14->15->16->17->18"	49
T5	"1->2->3->6->7->8->13->14->15->16->17->18"	30
T6	"1->2->3->6->7->8->9->10->12->13->14->15->16->17->18"	40
T7	"1->2->3->6->7->8->9->11->12->13->14->15->16->17->18"	40

When the above-mentioned test suite shown in Table 1 was executed by using GA program, the putative testing paths are generated:

T3: 1->2->3->4->5->6->7->8->9->10->12->13->14->15->16->17->18

T4: 1->2->3->4->5->6->7->8->9->11->12->13->14->15->16->17->18

T7: 1->2->3->6->7->8->9->11->12->13->14->15->16->17->18

When the above-mentioned test suite shown in Table 1 was executed by using FC program, the putative testing paths are generated:

T2: 1->2->3->4->5->6->7->8->13->14->15->16->17->18

T5: 1->2->3->6->7->8->13->14->15->16->17->18

T7: 1->2->3->6->7->8->9->11->12->13->14->15->16->17->18

When the above-mentioned test suite shown in Table 1 was executed using the proposed "GA-FC" program, the putative testing paths are generated:

T3: 1->2->3->4->5->6->7->8->9->10->12->13->14->15->16->17->18

T4: 1->2->3->4->5->6->7->8->9->11->12->13->14->15->16->17->18

The proposed "GA-FC" prioritized and minimized testing technique achieves 100 % ABC criteria whereas "GA" and "FC" achieves 66.66% and 33.33% respectively. Additionally, ETSM criteria achieves 71% by "GA-FC" while 57% achieves by "GA" and "FC".

3.2 Case Study-2 "Online Banking System"

The second case study is taken 'Online Banking' from[25]. Case Study-2 has three aspect nodes; authentication, authorization and logging. From the UML activity diagram of " Case study-2 has one aspect node: "Authorization". From the UML activity diagram of "Online Banking System" generated forty three independent testing paths[25]. Table 2 depicts the 43 independent testing paths and their PFVs are correspondingly for case study-2.

Table 2. Independent Testing Paths with their PFVs for case study-2

Test Case Id	Independent Test Paths	PFV
T1	{1,2,3,30}	11
T2	{1,2,3,4,5,7,8,9,11,12,13,15,16,17,30}	50
T3	{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,30}	71
T4	{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}	80
T5	{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}	81
T6	{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,20,22,28,30}	65
T7	{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,20,22,27,30}	64
T8	{1,2,3,4,6,7,8,9,11,12,13,15,16,17,30}	50
T9	{1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,19,21,23,24,30}	71
T10	{1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}	80

Continued on next page

Table 2 continued

T11	{1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}	81
T12	{1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,20,22,28,30}	65
T13	{1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,20,22,27,30}	64
T14	{1,2,3,4,5,7,8,10,11,12,13,15,16,17,30}	50
T15	{1,2,3,4,5,7,8,10,11,12,13,15,16,17,18,19,21,23,24,30}	71
T16	{1,2,3,4,5,7,8,10,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}	80
T17	{1,2,3,4,5,7,8,10,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}	81
T18	{1,2,3,4,5,7,8,10,11,12,13,15,16,17,18,20,22,28,30}	71
T19	{1,2,3,4,5,7,8,10,11,12,13,15,16,17,18,20,22,27,30}	64
T20	{1,2,3,4,6,7,8,10,11,12,13,15,16,17,30}	50
T21	{1,2,3,4,6,7,8,10,11,12,13,15,16,17,18,19,21,23,24,30}	71
T22	{1,2,3,4,6,7,8,10,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}	80
T23	{1,2,3,4,6,7,8,10,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}	81
T24	{1,2,3,4,6,7,8,10,11,12,13,15,16,17,18,20,22,28,30}	65
T25	{1,2,3,4,6,7,8,10,11,12,13,15,16,17,18,20,22,27,30}	64
T26	{1,2,3,4,5,7,8,9,11,12,14,15,16,17,30}	50
T27	{1,2,3,4,5,7,8,9,11,12,14,15,16,17,18,19,21,23,24,30}	71
T28	{1,2,3,4,5,7,8,9,11,12,14,15,16,17,18,19,21,23,24,25,26,29,30}	80
T29	{1,2,3,4,5,7,8,9,11,12,14,15,16,17,18,19,21,23,24,25,26,28,30}	81
T30	{1,2,3,4,5,7,8,9,11,12,14,15,16,17,18,20,22,28,30}	65
T31	{1,2,3,4,5,7,8,9,11,12,14,15,16,17,18,20,22,27,30}	64
T32	{1,2,3,4,6,7,8,9,11,12,14,15,16,17,30}	50
T33	{1,2,3,4,6,7,8,9,11,12,14,15,16,17,18,19,21,23,24,30}	71
T34	{1,2,3,4,6,7,8,9,11,12,14,15,16,17,18,19,21,23,24,25,26,29,30}	80
T35	{1,2,3,4,6,7,8,9,11,12,14,15,16,17,18,19,21,23,24,25,26,28,30}	81
T36	{1,2,3,4,6,7,8,9,11,12,14,15,16,17,18,20,22,28,30}	65
T37	{1,2,3,4,6,7,8,9,11,12,14,15,16,17,18,20,22,27,30}	64
T38	{1,2,3,4,6,7,8,10,11,12,14,15,16,17,30}	50
T39	{1,2,3,4,6,7,8,10,11,12,14,15,16,17,18,19,21,23,24,30}	71
T40	{1,2,3,4,6,7,8,10,11,12,14,15,16,17,18,19,21,23,24,25,26,29,30}	80
T41	{1,2,3,4,6,7,8,10,11,12,14,15,16,17,18,19,21,23,24,25,26,28,30}	81
T42	{1,2,3,4,6,7,8,10,11,12,14,15,16,17,18,20,22,28,30}	65
T43	{1,2,3,4,6,7,8,10,11,12,14,15,16,17,18,20,22,27,30}	64

When the above-mentioned test suite shown in Table 2 was executed by using GA program, the putative testing paths are generated:

T5:{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30},
T11:{1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30},
T17:{1,2,3,4,5,7,8,10,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30},
T23:{1,2,3,4,6,7,8,10,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30},
T35: {1,2,3,4,6,7,8,9,11,12,14,15,16,17,18,19,21,23,24,25,26,28,30},
T41:{1,2,3,4,6,7,8,10,11,12,14,15,16,17,18,19,21,23,24,25,26,28,30},
T4: {1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}, and
T10: {1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}

When the above-mentioned test suite shown in Table 2 was executed by using FC program, the putative testing paths are generated:

T3={1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,30}
T5={1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}
T2={1,2,3,4,5,7,8,9,11,12,13,15,16,17,30}
T6{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,20,22,28,30}
T11={1,2,3,4,6,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}

When the above-mentioned test suite shown in Table 2 was executed using the proposed "GA-FC" program, the putative testing paths are generated:

T5:{1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,28,30}
T4={1,2,3,4,5,7,8,9,11,12,13,15,16,17,18,19,21,23,24,25,26,29,30}

The proposed "GA-FC" and "GA" testing technique achieves 100 % ABC criteria whereas FC achieves 60%. Additionally, the proposed "GA-FC" achieves 95 % ETSM values whereas GA 79% and FC 88%.

3.3 Case Study-3 "On Line Shopping"

The third case study is taken 'Online Shopping' from [24]. The aspect weaving UML activity diagram for Case Study-3, in which new user first creates an account and registered with the website. The only registered user is allowed for subsequent interactions with the system [24]

Table 3. Independent Testing Paths with their PFVs for case study-3

Test case ID	Independent Test Paths	PFC
T1	{1, 2, 3, 4,5, 6, 16}	13
T2	{1, 2, 3, 4,5, 6, 7, 8, 9, 10, 16}	26
T3	{1, 2, 3, 4,5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16}	45
T4	{1, 2, 3, 4,5, 6, 7, 8, 9, 10, 12, 13, 14, 16}	47

When the above-mentioned test suite shown in Table 3 was executed by using GA program, the putative testing paths are generated:

T4:{1,2,3,4,5,7,8,9,10,11,12,13,15,16},

T3:{1,2,3,4,6,7,8,9,10,11,12,13,14,16},

T2:{1,2,3,4,5,6,7,8,9,10,16}

When the above-mentioned test suite shown in Table 3 was executed by using FC program, the putative testing paths are generated:

T4={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16

T2={1,2,3,4,5,6,7,8,9,10,16}

When the above-mentioned test suite shown in Table 3 was executed using the proposed "GA-FC" program, the putative testing paths are generated:

T4:{1,2,3,4,5,7,8,9,10,11,12,13,14,15,16}

T2:{1,2,3,4,5,6,7,8,9,10,16}

ABC criteria achieved 100% by "GA-FC" and "GA" testing techniques whereas "FC" achieves 66.66%. Similarly, ETSM is achieved 60% by "GA-FC" and 40% by "GA" and "FC".

Performance evaluation of all three case studies has been discussed in the next section 4.

4 Performance evaluation of "ga-fc" technique based on aop testing problem

This section discusses the results obtained when test suite minimization achieved using proposed "GA-FC" was comparatively evaluated against GA (Genetic Algorithm) and FC (Fuzzy Clustering) based techniques for all three case studies.

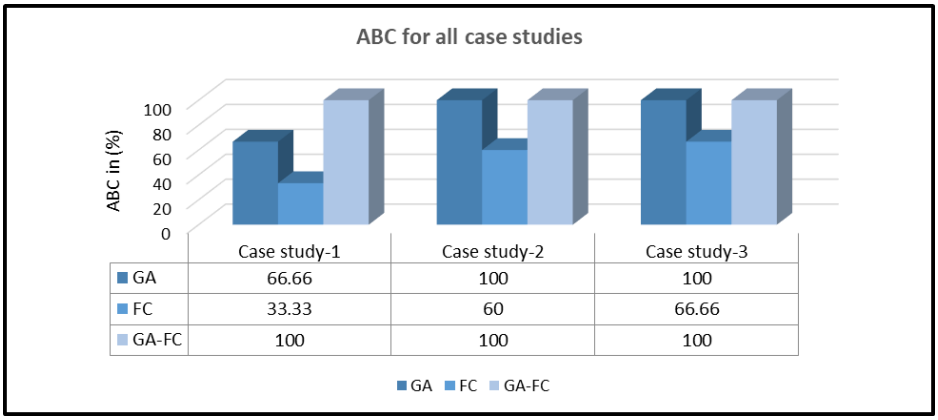

Fig 2. Comparative ABC graph for the proposed "GA-FC" technique against GA and FC

ABC percentage of the three case studies for each test suite minimization techniques namely, "GA-FC", GA and FC have been depicted by Figure 2 .

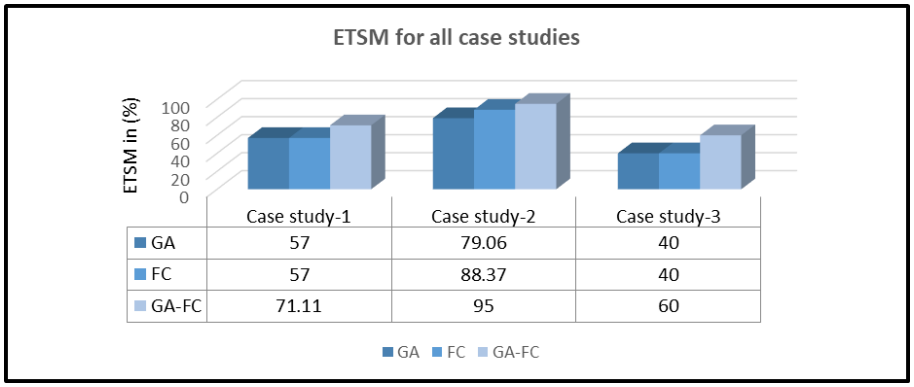

Fig 3. Comparative ETSM graph for the proposed"GA-FC" technique against GA and FC

Figure 3 depicts the comparison of percentage of test suite minimization for each case study by using proposed "GA-FC" technique along with GA and FC based test suite minimization techniques.

A comparative performance analysis of a novel "GA-FC" technique for test suite minimization for aspect-oriented software system with metaheuristic techniques namely, GA and FC. Figure 1 and figure 2 clearly describes the performance of "GA-FC" is either equally good or better than GA and FC. The AOSS problems taken in this study are three different computer science engineering case studies which have already been used earlier in a number of similar experiments.

5 Conclusion

The performance of the proposed "GA-FC" technique has been evaluated and compared with two soft computing optimization techniques namely GA and FC individually. The comparative results confirm that "GA-FC" performs either equally well or better than GA and FC techniques individually in terms of percentage of size of test suite reduction and percentage of complete aspectual branch coverage of reduced test suite. Moreover, any technique with higher test suite reduction percentage and maximum satisfies adequacy criteria improves the quality of the software as well as significantly reduced the time, effort and cost of the software system. Therefore, the proposed "GA-FC" technique proves its effectiveness with higher coverage of specified adequacy criteria (i.e. aspectual branch coverage criteria) using relatively smaller test suite. The future research will try to measure the performance of the "GA-FC" technique on real world problems using some other parameters.

References

1) Chauhan N. Software Testing Principles and Practices. 2nd ed. Oxford University Press. 2016.
2) Pressman RS, Maxim BR. Software Engineering: Practitioner's Approach. 8th ed.. 2019.
3) Jallote P. An Integrated Approach to Software Engineering. 2nd ed. Narosa Publishing House. 1998.
4) Chhabra R, Verma S, Krishna CR. A survey on driver behavior detection techniques for intelligent transportation systems. *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*. 2017;p. 36–47. Available from: https://doi.org/10.1109/CONFLUENCE.2017.7943120.
5) Boluwaji A, Gboyega D, Nwokoro C. A Systematic Review of Soft Computing Techniques for Software Testing. *International Journal of Computer Science & Management Studies*. 2019;40(1):7–17. Available from: https://www.researchgate.net/publication/336837415.
6) Jyoti, Hooda S. A Systematic Review and Comparative study of existing testing techniques for Aspect-oriented software systems. *International Research Journal of Engineering and Technology*. 2017;4(5):879–888. Available from: https://www.irjet.net/archives/V4/i5/IRJET-V4I5175.pdf.
7) Harrold MJ, Gupta R, Soffa ML. A methodology for controlling the size of a test suite. *Proceedings Conference on Software Maintenance 1990*. 1990.
8) Jain M, Gopalani D. Aspect-Oriented Approach for Testing Software Applications and Automatic Aspect Creation. *International Journal of Software Engineering and Knowledge Engineering*. 2019;29(10):1379–1402. Available from: https://dx.doi.org/10.1142/s0218194019500438. doi:10.1142/s0218194019500438.
9) Kaur C, Garg S. Testing Aspect-Oriented Software Using UML Activity Diagram. *International Journal of Engineering and Technology*. 2012;1(3).
10) Madadpour S, Hosseinabadi S. Testing Aspect-Oriented Software with UML Activity Diagram. *International Journal of Computer Applications*. 2011;(8):33–33.
11) Yoo S, Harman M. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*. 2012;22(2):67–120. Available from: https://dx.doi.org/10.1002/stv.430. doi:10.1002/stv.430.
12) Rhmann W, Zaidi T, Saxena V. Use of Genetic Approach for Test Case Prioritization from UML Activity Diagram. *International Journal of Computer Applications*. 2015;115(4):8–12. Available from: https://dx.doi.org/10.5120/20137-2232. doi:10.5120/20137-2232.

13) Srividhya J, Gunsundari R. Test Suite Minimization and Empirical Analysis of Optimization Algorithms. *Journal of Theoretical and Applied Information Technology*. 2016;(1):94–94.
14) Suresh Y, Rath SK. Genetic Algorithms Based Approach for Test Data Generation in Basis Path Testing. *Journal of Software Computing and Software Engineering*. 2013;3(3).
15) Xia C, Zhang Y, Hui Z. Test Suite Reduction via Evolutionary Clustering. *IEEE Access*. 2021;9(9):28111–28121. Available from: https://dx.doi.org/10.1109/access.2021.3058301. doi:10.1109/access.2021.3058301.
16) Harman M, Islam F, Xie T, Wappler S. Automated Test Data Generation for Aspect-Oriented Program. International Conference AOSD. 2009. Available from: https://doi.org/10.1145/1509239.1509264.
17) Biswal BN, Barpanda SS, Mohapatra DP. A Novel Approach for Optimized Test Case Generation Using Activity and Collaboration Diagram. *International Journal of Computer Applications*. 2010;1(14):67–71. Available from: https://dx.doi.org/10.5120/299-463. doi:10.5120/299-463.
18) S D, Hooda S. A Novel Approach for Testing an Aspect-Oriented Software System Using Prioritized-Genetic Algorithm (P-GA). *International Journal of Applied Engineering Research*. 2017;12(21). Available from: https://www.ripublication.com/ijaer17/ijaerv12n21_103.pdf.
19) S D, Hooda S. Test Suite Minimization using Fuzzy Clustering for Aspect-Oriented Software System. *IOSR-Journal of Engineering*. 2018;8(9):44–51. Available from: http://www.iosrjen.org/Papers/vol8_issue9/Version-2/H0809024450.pdf.
20) Delamare R, Kraft NA. A Genetic Algorithm for Computing Class Integration Test Orders for Aspect-Oriented Systems. *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. 2012;p. 804–813. Available from: https://doi.org/10.1109/ICST.2012.179.
21) Kumar G, Bhatia PK. Software testing optimization through test suite reduction using fuzzy clustering. *CSI Transactions on ICT*. 2013;1(3):253–260. Available from: https://dx.doi.org/10.1007/s40012-013-0023-3.
22) Kaur A, Grover PS, Dixit A. A framework for evaluating extensibility in Aspect-Oriented Software System and its Validation. *Recent Patents on Engineering*. 2020;14(4). Available from: https://doi.org/10.2174/1872212113666190625115111.2020.
23) Xie T, Zhao J. A Framework and Tool Support for Generation of Test Inputs to AspectJ Programs. . *Proceeding of the 5th International Conference on Aspect-Oriented Software Development*. 2006;p. 190–201. Available from: https://doi.org/10.1145/1119655.1119681.
24) Babu C, Krishnan HR. Fault model and test-case generation for the composition of aspects. *ACM SIGSOFT Software Engineering Notes*. 2009;34(1):1–6. Available from: https://dx.doi.org/10.1145/1457516.1457521.
25) Cui Z, Wang L, Li X, Xu D. Modeling and integrating aspects with UML activity diagrams. *Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09*. 2009;p. 430–437. Available from: https://doi.org/10.1145/1529282.1529377.
26) Dalal S, Hooda S. A Novel Technique for Testing an Aspect Oriented Software System using Genetic and Fuzzy Clustering Algorithm. In: 2017 International Conference on Computer and Applications (ICCA). IEEE. 2017.