Optimization of Test Cases in Regression Testing with Different Techniques.

Dhanashree Nankar[1], DR. Hemlata Patel[2]
[1]Student, MTECH, Parul University
[2]Faculty, PIET-MTECH, Parul University
Parul Institute Of Engineering And Technology

Abstract
The relevance of Software Testing (ST) in the Software Development Life Cycle (SDLC) should never be underestimated. Smarter ST can provide us with more reliable and defect-free goods that meet our stakeholder demands. That is why it requires more time and resources. Unfortunately, the product has reached the testing stage, particularly in Regression Testing (RT), with just a short time remaining.
Regression testing is one of the testing approaches used to ensure that changes made in repairs or other improvements do not impair the software's previously produced functionality. As a result, regression testing plays an important role in software testing. A generic software may change as a result of bug fixes, adaption to new environments, upgrading or updating functionality to increase performance, or as requested by a client.
After the programme has been provided to the customer, the generic software must be regressed to ensure that there are no flaws. Test case prioritisation aims to order test cases so that early issue identification is maximised. This paper examines each type of minimization, selection, and prioritisation strategy for future research.

Index Terms: Testing for Regression , Generic type of Software , Minimization of Test case

Introduction:

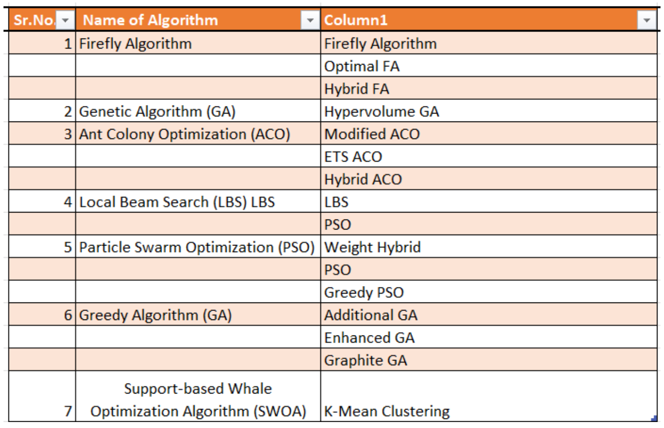
The practice for analysing and confirming that a software system or program meets its requirements and operates as intended is known as software testing. It entails running the software with numerous inputs to detect flaws or faults, assuring that it is dependable, secure, and works efficiently. Efficient software testing reduces the expense of fixing errors later in the development cycle and helps ensure that the customer receives a high-quality product. Software testing known as regression testing is used to confirm that recent changes to the code are not negatively impacting the features of an application. Regression testing is mostly used to find bugs that have been introduced into the software system or found in the functionality that has been in place as a result of recent changes like additions, patches, or configuration adjustments.

These days, software regression testing test mechanisation is impacted by data mining approaches. Regression testing would get more difficult as a result of the modifications in client requirements. Agile programming practices suggest a shorter software development life cycle and less restrictions on how regression testing can be carried out with constrained resources.

Purpose:

• Verify that new code modifications do not affect existing functionality.
• Ensure system performance remains consistent following changes.

When to Perform Regression Testing:
• After bug fixing.
• After introducing new features as well as enhancements.
• After performance enhancements and configuration tweaks.
Regression testing done right is essential to maintaining the integrity and quality of software throughout the course of its lifecycle, especially in agile development environments where continuous integration and frequent changes take place. Reviewing forty articles on Test Case Optimisation (TCO) published between 2010 and 2024 allowed us to undertake this survey. The different algorithms that have been investigated for TCO are displayed in Table 1.



Sr.No	Name of Algorithm	Column1
1	Firefly Algorithm	Firefly Algorithm
		Optimal FA
		Hybrid FA
2	Genetic Algorithm (GA)	Hypervolume GA
3	Ant Colony Optimization (ACO)	Modified ACO
		ETS ACO
		Hybrid ACO
4	Local Beam Search (LBS) LBS	LBS
		PSO
5	Particle Swarm Optimization (PSO)	Weight Hybrid
		PSO
		Greedy PSO
6	Greedy Algorithm (GA)	Additional GA
		Enhanced GA
		Graphite GA
7	Support-based Whale Optimization Algorithm (SWOA)	K-Mean Clustering

Table 1: TCO Algorithm

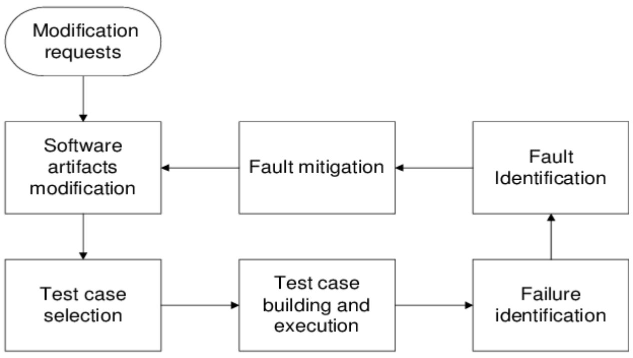


Fig 1: Regression Testing Process Model

Literature Review
1. Firefly algorithm
The flamboyant behaviour of fireflies served as the inspiration for the Firefly Algorithm (FA) [1].
• Because they are unisexual, fireflies are drawn to one another by nature.
• Firefly attraction depends on brightness. Both are precisely proportional. Fireflies move randomly if there is no brightness around.
Several algorithms have been proposed using the Firefly approach.

1.1. Optimal firefly algorithm

To find the optimal route, this program uses a metaheuristic firefly algorithm. By adding a objective like new function also the function for brightness and guiding matrices for graph traversal, [1] enhanced the firefly algorithm. Process is having first step creating data flow diagrams (DFD) and control flow graphs (CFG), then it generates XML files. The Optimal Firefly Test Sequence Generator receives this XML file (OFTSG).

1.2. Firefly algorithm

The use of function for fitness in metrices of string in the Firefly algorithm to find the best TCP configuration.

How this Algorithm for Firefly Works:

• Determine weight also distance for the test cases;
• Extract test cases from the dataset;
• Update movement based on the highest weight/distance values
• Select test cases with lowest distance as ideal sequence.

The optimal test case sequence was determined by selecting the move with the short distance and high weight over it across test cases. [1]

Six Software-artifacts Testing was conducted using Infrastructure Repository (SIR) programmes. Where we made in this method. Using cutting-edge algorithms such as Greedy and GA, FA outperforms LBS in certain areas. The following limitations were identified throughout the study.

1.3. Hybrid firefly algorithm (HFA)

[1] proposed a hybrid Firefly algorithm model that prioritises test cases and lowers testing expenses. Enhanced FA is added to a hybrid model, which HFA uses, once it has correlated all test cases and test data available. The dataset selection procedure is one issue our investigation found. This method uses three tiny SIR benchmark programs as its dataset. For larger test case spools, this approach might produce various outcomes.

2. Hypervolume genetic algorithm (HGA)

The Hypervolume Indicator and GA are used to rank test cases that satisfy more than three criteria. HGA (Hypervolume-based Genetic Algorithm), an enhanced version of the genetic algorithm, was developed by him. In the past, only single-objective optimisation problems were solved with GAs. Hypervolume is a treat that packs a lot of stuff into one. The two portions of HGA are as follows [6].

3. Ant colony optimization (ACO)

ACO also popular method in prioritising cases. Various ACO-based algorithms have been proposed. [6] introduced ant colony optimisation, a metaheuristic optimisation methodology. [12] employed standards based on requirements in their suggested ACO approach to address prioritisation issues. [12] separated the process in 2 steps: TCP-ACA-1, which calculates the distance within 2 test cases, and TCP-ACA-2, where continually updates the pheromone value. ACO achieved outstanding results in a variety of optimisation issues, including the Test Case Prioritisation problem [13]. employed an approach that the ACO algorithm functions in the same way it did from the beginning. The ants are randomly assigned to each node in the first list of test cases. The flaws have been identified. Which changes the pheromone value after every loop when it completed.

3.1. Modified ant colony optimization (m-ACO)

It [6] introduced a optimisation technique (m-ACO). Here, if an ant discovers food, it returns the food. The cycle of returning home and collecting food continues until all sources are exhausted. Additionally, ants tend to stick to a narrow trail until their food supply is depleted. The technique assessed code coverage, fault detection, and execution speed. [13] suggested the APFD metric for maximising fault coverage.

3.2. ETS-based ACO algorithm for TCP

Based on epistasis MoTCP, or Multi-objective Search-Based Regression Test Case Prioritization, is a technique used in Ant Colony Optimization. This technique is [13] describe the nature, such as the motion of ants in the search of food. According to [13], epistasis theory enhances the one goal at a time genetic algorithm. Positive outcomes were obtained when the multi-objective problem was solved using the epistasis theory.

E-ACO outperformed the other ACO algorithms discussed above. The novel methodology was compared to the cutting-edge NSGA-II algorithms in MoTCP also ACO, yielding impressive results. Although E-ACO produced significant results, it still has drawbacks that need to be addressed before it can become a widely adopted algorithm.
The first factor is instrumentation accuracy, which might lead to varied outcomes. It's better if we test this.
Instead than relying solely on the GCC tool gcov, consider using other tools.
To compensate for the absence of a real-time programme, a ten-fold test was conducted. Real-time large programmes provide more accurate analysis outcomes.
Use a variety of swarm sizes.

3.3. Hybrid ACO

In [1] developed a hybrid technique that combines two approaches.
Very firstly, factors for test are used in assign precedence to test instances.
Secondly, After assigning priorities in cases for testing , here use ACO in determine for ideal series with the fastest rate of execution also maximum error rate.
This paper lacks information on testing or comparisons with current approaches, despite its intriguing approach.

4. Local beam search algorithm (LBS) for TCP

[13] developed the LBS, which is a method of input-based search. In contrast to code-coverage techniques, the LBS method makes use of lightweight input. This algorithm is more efficient than other methods because it maintains a constant beam width throughout each iteration. [13] discovered that a greedy algorithm does not choose new test cases from previously discovered ones, resulting in higher costs and time. The input-based LBS algorithm processes at random chosen cases for test set to provide a prioritised set of test cases. [13] suggests utilising distance matrices to compare prioritised and unprioritized test instances. The LBS method gives outstanding results, but it has limitations.
1) The platform utilised for testing this strategy was merely a C. A programme was used. To cross-validate the findings, different programming languages should be investigated.
2) Suites for Testing: Just sufficient branch test suites were employed for this research. Using diverse test case suites can improve outcomes.
3) In addition to the distance metre, other metrics, such as the Hamming distance matrix, can be used to assess findings.
4) The LBS uses a randomised approach, thus the author does the tests again 50 times to improve findings. Using a large sample size of test cases is more appropriate.

5. Particle swarm optimization (PSO)

PSO also a multi-object TCP approach that prioritises cases for test according to code changes. PSO represents an optimisation. The approach was first introduced in [8] Souza prioritised test cases using the Binary Constrained PSO technique, which involves three phases [8].
• Remove unneeded test cases.
• Select test cases that provide optimal coverage while reducing execution time.
• Promoting test cases.
The study has the following limitations.
1) Consider internal validity. This approach was tested with 20 JUnit test cases, a tiny sample size.
2) Only the Greedy algorithm is used to assess the novel strategy's external validity. Verify this strategy with the help of a instantaneous implementation also compare it to other cutting-edge procedures for the best outcomes.

5.1. Weight hybrid TCP using PSO (WH-PSO)

[8] presented a weighted hybrid string distance algorithm. This strategy sought to achieve a greater APFD. We developed a hybrid string length that combines weight and separation between test instances. The distance between the strings is categorised into 2 types: character-based and term-based.
The technique was tried with 6 various card for JAVA programs (JCS Applets), including a network connection tracker and HelloWorld. PKI, RSA Cryptographic, Calculator, OATH, Passport, and CoolKey. GPSO surpassed the standard

genetic algorithm (GA) in a number of areas. GPSO involves less iterations and effort, but it delivers a high coverage percentage.
The limitations of this study are as follows.
1) The approach was tested using a real-time application with tiny programmes and few branches. In a real-time programme, thousands of code branches may provide distinct perspectives on the strategy.
2) The discussion regarding the fault matrix is missing.
3) While only one GA comparison was conducted, it is recommended to compare this novel strategy against other cutting-edge methods.

6. Greedy algorithm (GA)
6.1. Additional greedy algorithm (AGA)
As [8] presented the AGA. In this method employs NSGA-II and its variant, vNSGA-II algorithms. The vNSGA-II varies from the NSGA-II by two key changes:
• vNSGA-II provides different objectives to each sub-population, making it a useful method.
•vNSGA-II captures the best sub-population.
The study exposed the following flaws.
1) Tool accuracy when locating coverage information. To reduce risk, the author used professional software, however it is recommended that you try with various apps to see what results you get.
2) This study relies heavily on testing data from the repository for SIR.
Just 1 ESA programme were utilised.
3) Response of this technique is dependent on algorithm used for multi-object prioritising.

7. Support-based Whale Optimization Algorithm (SWOA)
This algorithm [7] employed regression testing approaches such as support-based Whale Optimisation and fractional sigmoid-based K-means (FSK-mean) modules. The cases for test evolved histories is use for making groups to reduce the number of test cases, prioritising test cases based on code coverage and clearing up any doubt in situations with the identical priority, and then selecting the test cases from each cluster are the main focus in order to identify the most flaws as soon as possible.
We examine results for Reduction of Test Cases and SWOA Optimisation for Distributed Agile Software Development with Testing for regression experimented.

Interpretation and analysis
This study gives an empirical survey of test case prioritisation methods. A systematic approach is used in this work to review 35 TCO papers published in journals and conferences.
The purpose of this research is to determine which approaches have received the most attention, which have been underutilised, and where there is space for improvement.
• On an average 60% of publications in this research employ a publicly available data source, which is encouraging. Unfortunately, the given information is not consistently evaluated, that is unsatisfactory. As programming languages and techniques improve, it becomes vital to update code for testing purposes. The usage of SIR data rather than updated public datasets looks to be gaining traction. However, there are barriers to acquiring authentic data and testing again by other investigators.
•Analysis of multi-object TCP approaches: We found that 35% of TCP techniques are multi-object, 20% are single-object, and the rest are not clearly characterised. It is suggested to develop a model-based multi-object technique to determine.
According to a poll, most research favours code base approaches over cost considerations. But just 10% of research

made use of non-SIR data sets. Conflict of interest and adherence to ethical principles. Here authors have disclosed there are no possible conflicts of interest related to the investigation, writing, or release of this work.

Conclusion and Future Scope

Optimising test cases in regression testing is critical for preserving software quality while lowering testing time and expenses. Techniques including test case selection, prioritisation, and minimization, as well as automation and machine learning, show promise for fast regression testing. Continued study and development in this sector are critical for addressing developing difficulties and improving the effectiveness of testing for regression systems. Here future scope of testing for regression is extensive, encompassing crucial sections of the product to ensure stability and dependability following modifications. By focusing on essential functionalities, new and modified features, UI, integration points, performance, security, compatibility, configuration changes, and backward compatibility, regression testing contributes to the software's quality and robustness.

References:

(1)Ahmad SF, Singh DK, and Suman P (2018). Prioritization for regression testing using ant colony optimization based on test factors. In: Singh R, Choudhury S, and Gehlot A (Eds.), Intelligent communication, control and devices: 1353-1360. Springer, Singapore, Singapore. https://doi.org/10.1007/978-981-10-5903-2_142
(2)Allawi HM, Al Manaseer W, and Al Shraideh M (2020). A greedy particle swarm optimization (GPSO) algorithm for testing real-world smart card applications. International Journal on Software Tools for Technology Transfer, 22(2): 183-194. https://doi.org/10.1007/s10009-018-00506-y
(3)Askarunisa MA, Shanmugapriya ML, and Ramaraj DN (2010). Cost and coverage metrics for measuring the effectiveness of test case prioritization techniques. INFOCOMP Journal of Computer Science, 9(1): 43-52.
(4)Azizi M and Do H (2018). Graphite: A greedy graph-based technique for regression test case prioritization. In the IEEE International Symposium on Software Reliability Engineering Workshops, IEEE, Memphis, USA: 245-251. https://doi.org/10.1109/ISSREW.2018.00014 PMid:30045707 PMCid:PMC6060527
(5)Bian Y, Li Z, Zhao R, and Gong D (2017). Epistasis based ACO for regression test case prioritization. IEEE Transactions on Emerging Topics in Computational Intelligence, 1(3): 213-223. https://doi.org/10.1109/TETCI.2017.2699228
(6)Di Nucci D, Panichella A, Zaidman A, and De Lucia A (2015). Hypervolume-based search for test case prioritization. In: Barros M and Labiche Y (Eds.), International symposium on search based software engineering: 157-172. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-22183-0_11
(7)Madan Singha*, Dr. Naresh Chauhan b, Dr. Rashmi Poplic a Research Scholar at J. C. Bose University of Science and Technology, YMCA Faridabad, India. b Professor in Department of Computer Engineering at JC Bose University of Science & Technology, Faridabad, India." Test Case Reduction and SWOA Optimization for Distributed Agile Software Development Using Regression Testing" (2023)
(8)ARPN Journal of Engineering and Applied Sciences ©2006-2015 Asian Research Publishing Network (ARPN)." STRUCTURAL SOFTWARE TESTING: HYBRID ALGORITHM FOR OPTIMAL TEST SEQUENCE SELECTION DURING REGRESSION TESTING"(2015)
(9) Regression testing minimization, selection and prioritization:a survey M. Harman King's College London, Centre for Research on Evolution, Search and Testing, Strand, London WC2R 2LS, U.K.
(10) 2011 27th IEEE International Conference on Software Maintenance (ICSM)
Regression Testing in Software as a Service :An Industrial Case Study Hema Srikanth IBM Lotus Division Littleton, MA

(11) International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-10, August 2019 "Regression Testing of Service-Oriented Software " Kaushik Rana, Harshal Shah, Chetan Kapadiya
(12) Test case prioritization techniques in software regression testing: An overview Muhammad Qasim 1, Asifa Bibi 2, Syed Jawad Hussain 3, N. Z. Jhanjhi 4, Mamoona Humayun 5, Najm Us Sama 6.

(12) Nayak S, Kumar C, Tripathi S, and Jena L (2019). Efficiency enhancement in regression test case prioritization technique. International Journal of Innovative Technology and Exploring Engineering, 8(12): 5445-5451. https://doi.org/10.35940/ijitee.K1595.1081219
(13)Yuan F, Bian Y, Li Z, and Zhao R (2015). Epistatic genetic algorithm for test case prioritization. In: Barros M and Labiche Y (Eds.), International symposium on search based software engineering: 109-124. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-22183-0_8
(14) Image for regression process model from Research Gate Link:- Process of regression testing | Download Scientific Diagram (researchgate.net)