International Journal of Industrial Engineering & Production Research December 2023 Vol. 34, No. 4: 1-14
DOI: 10.22068/ijiepr.34.4.7

RESEARCH PAPER

The State-of-the-art Hybrid Approaches in Regression Testing

Muhammad Asim Siddique[1*], Wan M.N Wan-Kadir[2] & Johanna Ahmad [3]

Received 21 June 2023 ; Revised 17 July 2023; Accepted 20 August 2023;

ABSTRACT

Software testing is the process of assessing the functionality of a software program. The software testing process checks for inaccuracies, gaps and whether the application outcome matches desired expectations before the software is installed and goes into production. Normally in large organizations, the development team allocates a high portion of estimated development time, cost and efficiency for regression testing to assure software testing quality assurance. The quality of developed software relies upon three factors time, efficiency and testing technique used for regression testing. Regression testing is an important component of software testing and maintenance, taking up a significant share of the total testing time, efficiency and resources organizations use in testing techniques. The key to successful regression testing using Test Case Prioritization (TCP), Test case Selection (TCS) and Test Case Minimization (TCM) is maximizing the test cases' effectiveness while considering the limited resources available. Regression testing introduced numerous techniques for (TCP, TCS, TCM) to maximize the efficiency based on Average Percentage Fault Detection (APFD). In recent studies, the TCP and TCS techniques can give the highest APFD score. However, each TCP and TCS approacshow limitations, such as high execution cost, time, efficiency, and lack of information. TCP and TCS approaches that can cover multiple test suite variables (time, cost, efficiency) remained inefficient. Thus, there is a need for a hybrid TCP and TCS technique to be developed to search for the best method that gives a high APFD score while having good coverage of test cases relevant to the cost and execution time to improve efficiency. The proposed hybrid test case selection and prioritization technique will exclude similar & faulty test cases to reduce test size. The proposed hybrid technique has several advantages, including reduced execution time and improved fault detection ability. The proposed hybrid Enhanced Test Case Selection and Prioritization Algorithm (ETCSA) is a promising approach to select only modified test suites to improve efficiency. However, the efficiency of the proposed technique may depend on the specific criteria for selecting only modified test cases and the software's characteristics. The hybrid technique aims to define an ideal ranking order of test cases, allowing for higher coverage and early fault detection with reduced test suite size. This study reviews TCP and TCS hybrid techniques to reduce testing time, cost and improve efficiency for regression testing. Each TCS and TCP technique in regression testing has identified apparent standards, benefits, and restrictions.

KEYWORDS: *Regression testing;software testing; Test-case prioritization; Test case selection; Test case minimization; Hybrid techniques.*

1. Introduction

Software testing is essential to the process of developing and maintaining software since it helps to ensure the reliability and quality of the end product.[1]. Regression testing is an important component of software testing, often involving the execution and evaluation of changes in code or the environment that could potentially affect the correct functioning of the program. This form of testing is repeated multiple times, often after each shift in the program, to maintain its proper functioning [2].

Regression testing is an important technique used by software companies to ensure the quality of their products. It is used to verify the correctness of the software whenever changes are made and

* **Corresponding author:** ***Muhammad Asim Siddique***
muhammadsiddique@graduate.utm.my

1. *Faculty of Computing, Universiti Teknologi Malaysia, Skudai, Johor Baharu, 81310, Malaysia.*
2. *Faculty of Computing, Universiti Teknologi Malaysia, Skudai, Johor Baharu, 81310, Malaysia.*
3. *Faculty of Computing, Universiti Teknologi Malaysia, Skudai, Johor Baharu, 81310, Malaysia.*

before releasing a new version to the public. However, regression testing requires significant resources and is time-consuming due to the need to re-execute all test cases of the previous version to ensure that the changes made do not cause any unexpected issues. Therefore reducing the test suite size and selecting only the most relevant and necessary test cases becomes essential [2].

Regression testing aims to detect errors introduced by changes to the code or the environment without executing all tests necessary to cover the entire system. To do this, testers must prioritize certain test cases over others and select which tests are most likely to detect errors quickly. However, while prioritizing and determining the precise tests can help increase the effectiveness of regression testing, the process can become quite time-consuming and costly.

Programming and constantly updating software are critical to today's technological advancement. When software is delivered to clients, it faces new obstacles, such as satisfying unforeseen client requirements, unexpected inputs, increased market competitiveness, and changing user/client demands [3]. These issues must be resolved once the software has completed its development phase and is in its maintenance phase. If not, the programme is rendered useless. It is either necessary to develop new software, which is very expensive in terms of computational time and resources, or it is necessary to update old software. Software updates are subject to definite budgetary and time restraints as well as pressure to achieve the required change objectives. To maintain its precision and accuracy after an update, the programme must be tested again [3].

Regression testing's primary objective is to ensure that changes haven't negatively impacted the functioning of SUT components that are already in use, resulting in an architecture that is clear-cut, precise, scalable, effective, safe, and reliable[4]. Almost all software contains problems because humans who write it make mistakes. Almost all software contains problems because humans who write it make mistakes. This is inevitable, and there are other risks as well, such as requirements that are unclear or incorrect, requirements that are misunderstood, software components that are misused, errors made by developers when coding, and even code that was previously working but may now be incorrect due to assumptions that were once valid but are no longer true after changes. Software testing is a clear answer to this problem, and several study areas need to be considered in order to create a strong research framework. The existence of TCS, TCP, and TCM approaches, hybrid approaches, similarity algorithms, ranking algorithms, history-based selection approaches, and other regression testing methodologies should all be considered in all activities. Software testing takes a lot of time and money. The sheer quantity of potential tests for any nontrivial system is one major factor in this. There are now 232 potential tests for a straightforward architecture that accepts a single 32-bit integer number as input. Which of these must be completed in order for this study to find every bug? Unfortunately, the answer to this is that the study will have to run every test, but only if we run them all. It is clear from this that running every test does not scale. Dijkstra's famous remark that testing can never ensure the absence of flaws effectively captures this truth. It can only reveal that there are bugs present. Finding a subset of the potential system inputs that will provide us with an acceptable level of assurance that the big problems were discovered is thus the difficult task[1].

The acquisition and evaluation of software quality depend on software testing. Despite all the advancements in programming languages and software development methodologies, software testing remains crucial. The purpose of testing is to ensure that the software artefacts under consideration function as intended and do not take any unintended actions, therefore enhancing the quality of these items. However, testing is expensive, resource-intensive, and notoriously difficult: According to surveys, testing costs more than half the overall cost of developing software.[5]. Additionally, testing is prone to error as with any human-driven activity, and developing trustworthy software systems is still a challenge. Researchers and practitioners have been looking towards more efficient ways of testing software in an effort to address this issue.

According to current market trends, regression testing is essential for the proper operation of software products. Within the software development life cycle, it is an expensive and time-consuming ongoing procedure. Each test case is assigned a selection probability based on its capability to accomplish a specific testing objective in a test case prioritization. To improve testing performance, it chooses which test cases should be run in priority order. Techniques for prioritising test cases could be very helpful in boosting test suite effectiveness in actual use. The technique of test case prioritising aids in improving the rate of fault finding. In contrast, Regression test prioritisation is frequently done in a time-constrained execution environment where testing is limited to a set amount of time.[6].

Regression testing [2] is crucial to software enhancement since it ensures that the updated

alterations don't disrupt the software's current functionality. However, due to the numerous test cases and revisions in the current application, it is extremely important and expensive to run the whole set of regression test suites. Regression testing is typically used by the quality assurance team to identify a substantial portion of sophisticated communicated frameworks. This is on the grounds that: (1) building and raising different segments on servers is costly, (2) an experiment will in general, take additional time, precision, and assets to keep running because of system idleness, and (3) distinct updates of parts can likewise make more blends to test.

Regression testing (RT) is a re-testing procedure to make sure that system changes don't have an impact on other components and unrelated aspects continue to be stable as before. [7]. As shown in (Fig. 1), Regression testing generally uses the following three techniques: test case minimization (TCM), test case selection (TCS), and test case prioritisation (TCP)[8]. Test cases that are deemed redundant or nearly identical will be reduced or deleted using the TCM approach. The test cases that satisfy the precise requirements will be chosen by TCS. In the interim, TCP will prioritise the test cases using the stated criteria [9]. There is a need for greater research since the regression testing research field faces the problem of improving effectiveness. [10].

2. Test Case Prioritization (TCP)

Choosing which tests to run within a specific testing session is referred to as test case selection (TCS) in regression testing.[14]. This procedure entails assessing the application's present state, the modifications that have taken place, and the goal of the regression testing session. All the areas that could be influenced by the changes, such as functionality, performance, reliability, and usability, should be covered by the tests that are chosen. To cut down on time during test case selection, regression testing must be set up with numerous variations.[14].

A effective RT technique involves finding and selecting only the appropriate updated test cases in the running programme that have been touched by code updates. Test case selection in regression testing is a helpful technique for assuring the accuracy of modified versions while reducing testing costs. Because a small number of test cases will retest the existing software, a "regression test selection" process is used to ensure that the changing application components are not interfering with the unchanged elements[15]. Regression testing is an expensive and popular maintenance technique used to confirm updated software. Test case selection (TCS) solutions deliberately pick a smaller portion of the test suite in an effort to reduce expenses without noticeably affecting effectiveness. Some even promise that the chosen test cases won't perform worse than the original test suite in certain circumstances. However, this fails to consider key realities of software development, such as resource and time limitations, which may preclude the correct deployment of TCS approaches (for instance, regression testing must be completed over the course of an entire night, but TCS selection only returns tests for the previous two days). Test Case Selection (TCS) aims to pick and execute just those tests that are affected by code changes because tests that are not should have the same results as earlier runs. In this approach, TCS, which is widely used in real-world applications, can dramatically minimise regression testing[16]. The two dimensions of information needed by a typical TCS technique are the altered programme components and the test dependency information, which refers to the programme elements that can be run during each test execution. Then, using a secure TCS technique, any test whose dependencies coincide with the modified programme pieces are chosen as the affected tests because not running any tests could result in any regression bugs being undetected[17]. In actuality, testers get around this by prioritising the test cases and only running those that comply with the limitations that already exist. Sadly, this frequently violates crucial TCS presumptions, nullifying TCS approach guarantees and unpredictable regression testing performance. However, current prioritisation methods are memoryless, relying on local decisions to provide appropriate long-term performance[15].

2.1. Test case selection (TCS)

Choosing which tests to run within a specific testing session is referred to as test case selection (TCS) in regression testing. [14]. This procedure entails assessing the application's present state, the modifications that have taken place, and the goal of the regression testing session. All the areas that could be influenced by the changes, such as functionality, performance, reliability, and usability, should be covered by the tests that are chosen. To cut down on time during test case selection, regression testing must be set up with numerous variations.[14].

A good RT technique involves identifying and selecting only the updated test cases that are relevant within the current programme that have been influenced by code updates. Test case selection in regression testing is a helpful

technique for assuring the accuracy of modified versions while reducing testing costs. Because a small number of test cases will retest the existing software, a "regression test selection" process is used to ensure that the changing application components are not interfering with the unchanged elements. [15]. Regression testing is a pricy and well-liked maintenance method for verifying upgraded software. By carefully selecting a smaller portion of the test suite, test case selection (TCS) solutions aim to reduce expenses without significantly affecting efficacy. Some even promise that the chosen test cases won't perform any worse than the original test suite in certain circumstances. However, this fails to take into account key realities of software development, such as resource and time limitations, which may preclude the correct deployment of TCS approaches (for instance, regression testing must be completed over the course of an entire night, but TCS selection only returns tests for the previous two days). Test Case Selection (TCS) aims to pick and execute just those tests that are affected by code changes because tests that are not should have the same results as earlier runs. Regression testing can be significantly reduced in this way thanks to TCS, which is frequently utilised in real-world applications[16]. The two dimensions of information needed by a typical TCS technique are the altered programme components and the test dependency information, which refers to the programme elements that can be run during each test execution. Then, using a secure TCS technique, any test whose dependencies coincide with the modified programme pieces are chosen as the affected tests because not running any tests could result in any regression bugs being undetected[17]. In actuality, testers get around this by prioritising the test cases and only running those that comply with the limitations that already exist. Sadly, this frequently violates crucial TCS presumptions, nullifying TCS approach guarantees and unpredictable regression testing performance. However, current prioritisation methods are memoryless, relying on local decisions to provide appropriate long-term performance [15].

2.2. Test case minimization (TCM)

Finding which existing tests should be preserved and which can be dropped while still providing enough coverage of the code under test is known as test case reduction (TSM). In most cases, this entails examining the current test suite to determine which tests are redundant and which are pertinent. Additionally, it guarantees that the remaining tests cover the code or functionalities under test. Another definition of Test Suit Minimization (TSM) is the inclusion of all test cases that can identify flaws while excluding all test cases that cannot do so from the final test suite. All of the test cases that can or cannot disclose faults during regression testing are included in the original test suite. The quality testing criteria and terms listed above are those that must be realised and met in order to ensure quality testing[18].

3. Comparison of Regression Testing Techniques

Using characteristics like approach, strength, weakness, and improvement, Table 1 compares the three different regression testing approaches to see how they differ from one another. In comparison to TCS and TCM, Test Case Prioritization is the most efficient method for optimising early defect detection [19]. Furthermore, neither minimization nor selection order test cases; instead, they aim to reduce the number of test sets. The goal of test case prioritisation is to rearrange test cases according to priority while attempting to maintain the size of the test suite. The trials' results showed that combining techniques produces better results than employing information retrieval and code coverage separately [20]. They were able to increase the regression testing coverage in a controlled trial, which makes their suggested method more productive. They used a hybrid strategy (TCP and TCS) for their case study, which improved the regression testing outcome. It is thus demonstrated that the hybrid strategy not only resulted in fewer test cases, lower costs, and faster execution, but also in a high rate of fault detection and enormous scalability [21].

Tab. 1. Comparison of regression testing techniques

Authors	Technique	Approach	Strength	Weakness	Improvement
(Panda and Mohapatra, 2017)[22]	Minimization	Each test case undergoes the framework and selection made according to the control flow diagram	Testing cost is reduced	The rate of fault detection decreases	Testers are versatile in the choice of standard test cases, depending on their budget
(Arrieta, Wang, Markiegi, et al., 2018)[23]	Selection	Applied black-box metrics with machine learning and experimented on a simulation model	Retest for updated system work, and time was reduced	High cost as testing required to compute power to learn test cases behavior	The selection approach in finding similarity has given a cost-effective result
(S. R. H. S. Kazmi, 2017)[24]	Selection	Applied a weight-based approach to validate and refine test suites	Regarding inclusivity and precision metrics, all of the situations used operate in a reasonably acceptable manner.	The designed framework does not accommodate code change information or requirement change	A smaller test suite, lower cost, more thorough testing, the capacity to discover faults, and information on code changes
(Jahan et al., 2020)[25]	Prioritization	Applied risk-based approach by prioritizing several factors, which are requirement modification information, complexity, and size of requirement	A high rate of fault detection and higher APFD in high-risk modules	Not suitable for large-scale system	Faster identification of risky faults in high-risk module

3.1. Hyrid approaches

Regression testing hybrid approaches combine various testing methodologies to make sure that software updates do not result in new bugs or regressions in previously tested functionality[37]. Here are some instances of hybrid regression testing techniques. Combining exploratory testing with automation testing This method covers the majority of the regression test cases with automated tests, while exploratory testing finds any new flaws or problems that automated tests might have overlooked. Combining risk-based testing with ad-hoc testing: In this method, regression tests are prioritised according to the risk involved in making software modifications. Automated tests are used to test high-risk areas and ad-hoc tests are used to test low-risk areas. User-based testing in addition to code-based testing: While user-based testing examines the system from the viewpoint of the end user, code-based testing examines the code directly. In this approach, the usability and user experience of the system are assessed using user-based testing, and the core functionality is assessed using code-based testing.The coverage-based TCP technique is the one that was first discussed in the TCP study and has since been used extensively [19]. The detection of all potential system flaws during regression testing is theoretically possible with 100% system coverage. However, in order to attain 100 percent coverage, a significant amount of testing time and more resources would be needed, which would add to the overall time necessary. This explains why academics seem to use that methodology in their analysis. In 2020, [20] suggested a method for selecting and

prioritizing test cases for regression that is both efficient and effective. They aim to develop a high-coverage Test Plan that will result in the most defects being discovered. Figure 1 explains the classifications of hybrid approaches for test case prioritization (TCP) and test case selection (TCS). Using information retrieval and code coverage separately, the results of the experiments demonstrated that the combined technique produces better outcomes than using them together. Finally, based on their observations, it has been shown that a combined approach delivers better performance rather than separately expanding information retrieval or code coverage. [30] also applied coverage based on their regression testing. TCP, TCS, TSM, and a hybrid strategy that combines selection and reduction were the four main techniques that were examined. Their investigation revealed that the prioritisation method combined well with the finer-grained coverage criterion. Additionally, TSM with more precise coverage criteria usually
achieves notable execution cost savings of 79.5 percent while keeping a fault detection efficiency. above 70 percent. The outcomes, however, also demonstrated that the hybrid approach did not considerably outperform the conventional minimization techniques.

In a similarity-based approach, variables like requirements, input strings, and test cases are compared to see how similar they are. The strategy's objective is to increase the variety of chosen test cases, which is determined by similarity metrics like string distance. As a result, the enhancement in the diversity of the test cases will have a better chance of identifying bugs as early as possible [38]. Similarity-based work can be seen in these publications [39] [40]. Both forms of research used the same methodology and strategy. They utilised the clustering method to put test cases in groups to discriminate between fault-revealing and non-fault-revealing test cases in a test suite. Their findings demonstrated that the test suite size was greatly decreased by the cluster selection technique. As a result, regression testing becomes less expensive[40]. This demonstrates the substantial impact that this strategy has. A hybrid approach in regression testing aims to integrate the benefits of various testing methods in order to raise the overall standard and efficiency of the testing procedure.

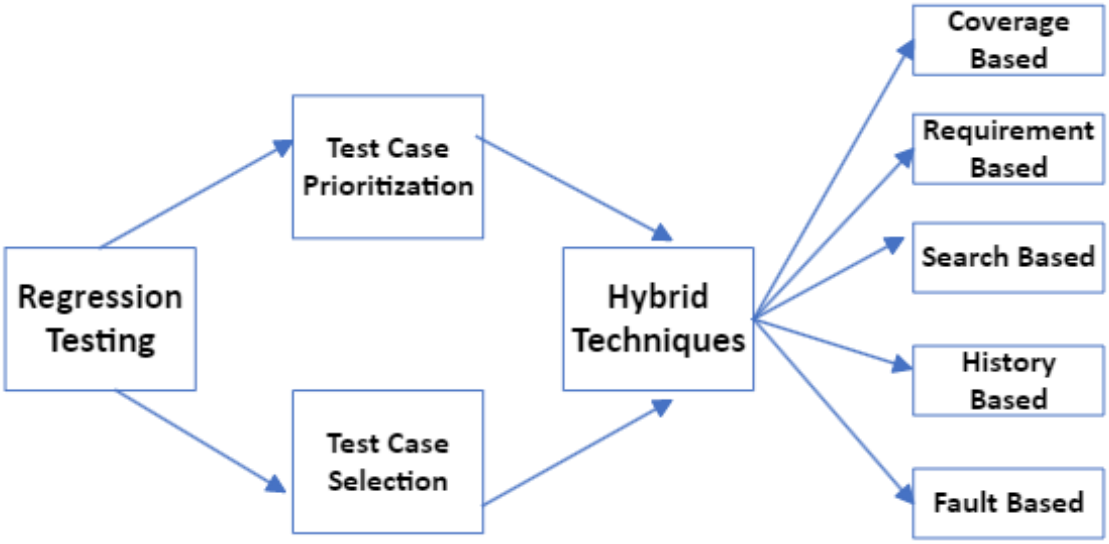

Fig. 1. Classification of hybrid approaches

3.2. Comparison of hybrid studies

The comparison among all existing hybrids is shown in Table II. The first column shows the name of the researcher involved in the study. The second column is for the study's primary aim. The third one is meant for the technique used in their research, which could be TCS, TCP or TSM. The fourth column is the approach used by the researchers. The last column summarizes their findings regarding the technique implementation.

Tab. 2. Comparison of hybrid studies

Authors	Objective	Hybrid Strategy	TCP Approach	TCS Approach	TSM Approach	Summary

(Bajaj and Sangwan, 2021)[43]	To minimize costs and work while optimizing the testing's efficiency and effectiveness	TCP TCS TSM	Code-coverage Method: Particle Swarm Optimization (PSO) & Gravitational Search Algorithm (GSA)	History-based Method: None	Code-coverage	1. Change-based is relevant for a small portion of changes 2. The research makes a substantial boost in overall test time 3. TCP strategy has proven
(Magalhães et al., 2020)[20]	To provide a comprehensive test plan that maximizes fault detection capability	TCS TCP	Risk-based	1. Information Retrieval 2. Code coverage Method: a. total coverage b. additional	None	The combined technique outperforms the independent usage of information
(Ali et al., 2019)[21]	To validate the effectiveness of the hybrid technique	TCP TCS	Code-coverage Method: Clustering	Risk-based -Identify highest priority in each cluster	None	1. Test suite reduced 2. Execution time reduced. 3. Correctness achieved is very high 4. Faults were Detected
(Su et al., 2020)[44]	To propose a hybrid model where it can reduce test cost	TCS TCP	Similarity-based	Similarity-based Method: The string edits distance and	None	The hybrid model improved test efficiency while lowering
(Shi et al., 2015)[42]	Cutting expenses by skipping some of the tests in the test suite and evaluating the effectiveness of	TSM TCS	None	Requirement-based - Select changes requirement	Code-coverage - Remove redundant test cases	Provides more significance. Savings in the number of tests but with a possible loss in fault detection
(Alves et al., 2013)[45]	To obtain a high rate of fault detection	TCS TCP	Coverage-based	Fault-based - Discover faults - Select test cases impacted towards the faults	None	1. Better and more stable results compared to traditional priority methods 2. Efficient in the early detection phase leading to risk and cost

Table II clearly shows that TCS and TCP are the best combos compared to TSM and TCS. Conclusion [41] has shown that it can reduce many test cases without losing the capacity to detect defects. sides that, a summary provided by [29] showed that the reduction technique could not produce an optimum result with other techniques. Another discouraging report is from [42] works, which also showed unsatisfactory results from a hybrid technique that combines TSM and TCS. The result gave more significant savings in the number of tests but with a possible loss in fault-detection. Even though there are few unsatisfied summaries, positive reports all came from studies where TCP and TCS were applied together. To further support the hypothesis, Wenjun et al.

(2020) work on the hybrid model also shows a promising result where test case size and test cost are reduced. (Magalhães et al., 2020)[20] has used risk based hybrid approach to maximize fault detection and (Su et al., 2020)[44] used similarity based method to remove the redundancy. Our approach will enhance both these approaches and propose a hybrid approach to improve efficiency in regression testing.

4. Selected Regression Testing Studies for Comparative Analysis

Regression testing is essential for the smooth operation of the System Under Test (SUT) and its testing since it has the potential of efficient incorporation and procedure for cost and precision. The difficulties with the efficiency strategies currently used for prioritising regression test cases are shown in Table III.

5. Evaluation of Hybrid Techniques

The proposed ETCPM technique will perform efficiently and is cost-effective compared to other techniques. The hybridization of the two same techniques helps produce an efficient solution for the execution of testing that detects more faults earlier and covers a reasonable size of a test suite. Besides, the proposed technique is also beneficial for the test engineer as the technique considers APFD and coverage test cases as experiment measurements, which gained more information on the test suite before executing regression. The most popular method used for Fault detection is APFD and APFDc. Both these methods will be used to evaluate the ETCPM hybrid techniques.

5.1. Comparison of hybrid studies

The weighted average of the faults found is referred to as the APFD metric. APDF is a frequently used statistic for assessing test case prioritisation strategies. The APFD metric was calculated [52] to assess the fault detection efficiency of coverage-based prioritisation methods depending on the fault criterion taken into account. APFD determines the weighted average of the percentage of defects discovered during a test suite. A higher number simply denotes a higher rate of fault detection; the APFD has a range of 0 to 100. [53]. Let T be a test suite containing n test cases, and let F be a set of m faults revealed by T. Let TFi be the first test case in ordering T0 of T, which reveals fault i. The APFD for test suite T0 is given by the equation as: -

$$APFD = 1 - \frac{TF1+TF2+\cdots+TFm}{nm} + \frac{1}{2n} \quad (1)$$

5.2. Average percentage of fault detected with cost (APFDc)

The modified APFD, or APFDc, includes the costs of faults. The resources needed to perform and validate the test case directly affect the cost of the test case. Several actions are available, including: Test cost can be calculated as the real time necessary to execute a test case when a machine or human is the main resource that is required. Another metric takes into account the monetary costs of executing and validating test cases, which may include hardware costs, labour costs, the cost of test-related materials, etc.. The APFDc metric is calculated based on the cost criterion that this study took into consideration.[54]. Let T represent a test suite with n test cases and costs.t1, t2…,tn. Let F be a set of m faults revealed by Ti and let f1, f2,…., fm be the severities of those faults. Let TFi be the first test case in an ordering T' of T that reveals fault i. the (cost-cognizant) weighted average percentage of faults detected during the execution of test suite T' is given by an equation as:-

$$\text{APFDc} = \frac{\sum_{i=1}^{m}(fi*(\sum_{i=TFi}^{n} ti-\frac{1}{2}t_{TFi}))}{\sum_{i=1}^{n} ti*\sum_{i=1}^{m} fi} \quad (2)$$

Tab. 3. Comparison of hybrid studies

Approach	Description	Remarks	Limitation / Weaknesses in the existing approach	Strengths of existing approaches
History-based TCP method (Wang, 2016)[46]	History-Based Dynamic Test Case Prioritization for Requirement Properties in Regression Testing	Based on previous research, this paper suggests a method for ranking test cases in order of importance. The testing procedure heavily depends on the requirements. In a history-based method, test case priorities are initially determined	The suggested approach has some drawbacks. When faults are broken down into each property need, there could be redundant faults. Redundancy reduction could be the top priority in future efforts. Large-scale experiments must be conducted.	This initialization technique is more effective than the random technique. Our history-based TCP method has the best efficacy and fault-detection capabilities when it comes to regression testing, according to experimental results, which differ significantly from those of the other methods.

		based on requirement priorities, based on historical data.		
Unival (Ammar, 2018)[47]	The Unival enhanced weighted technique prioritises test cases by randomly sorting test cases when two or more test cases record equal priority ratings.	Unival prioritises test cases based on code coverage requirements and information from the history of preceding runs by assigning each test case a unique priority value. The last execution's history and code coverage metrics are utilised to rank TC. Efficacy as determined by the typical coverage rate.	The trials were carried out in a constrained setting on a modest scale without consideration for time or expense. If the test is conducted on a medium to big size, the findings may differ..	Results show that this strategy performs better than other ways stated in the study in terms of prioritising test cases and achieving higher APFD values.
Fast (Miranda, 2018)[48]	FAST Approaches to Scalable Similarity-based Test Case Prioritization	Scalable similarity-based test case prioritising is offered by FAST approaches in both white-box and black-box settings.	Time and precision Evaluation criteria not mentioned clearly	Experiments performed with a high recall ratio and considerable accuracy percentage
ANT Algorithm with Faults Severity (Vescan, 2022)[11]	Test Case Prioritization—ANT Algorithm with Faults Severity	This paper suggests an ANT-based optimization algorithm with several criteria, including the number of covered defects, execution cost, and fault severity..	Future work could improve the TCP-ANT by using beneficent metrics (related to the testing type) for large test case suites.	To find the most faults with the greatest severity and shorten the regression testing period, the average percentage of fault identified metric is utilised as the best selection criterion..
Ant Colony Optimization (ACO),(Gao, 2015)[49]	Regression Testing Test Case Prioritization Using Ant Colony Optimization	This method saves time while improving fault detection.	Only white box testing is permitted for the strategy, and trials are only permitted to have 5–10 test cases..	Prioritises the test cases based on (ACO), accounting for the number and nature of problems discovered as well as the time required for execution..
Multi-Armed Bandit (MAB) (Lima, 2022)[50]	A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments	Compared to other methods, more efficient in terms of early fault identification and performance	The not specified quality attribute in the experiment, Approach is introduced in a very general way.	A comprehensive approach with enough experiments.
Adaptive random sequences (ARSs) (Chen, 2018)[51]	Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering	Using the K-means and K-medoids clustering algorithms, test cases are clustered in this study based on the number of objects and techniques.	This study needs to improve its sampling methodology in order to optimise the TCP test cases more successfully. Additionally, the quantity of test cases needs to be increased.	The study's findings indicate that the method coverage and random prioritisation TCP techniques are more successful and have a higher possibility of detecting faults earlier.

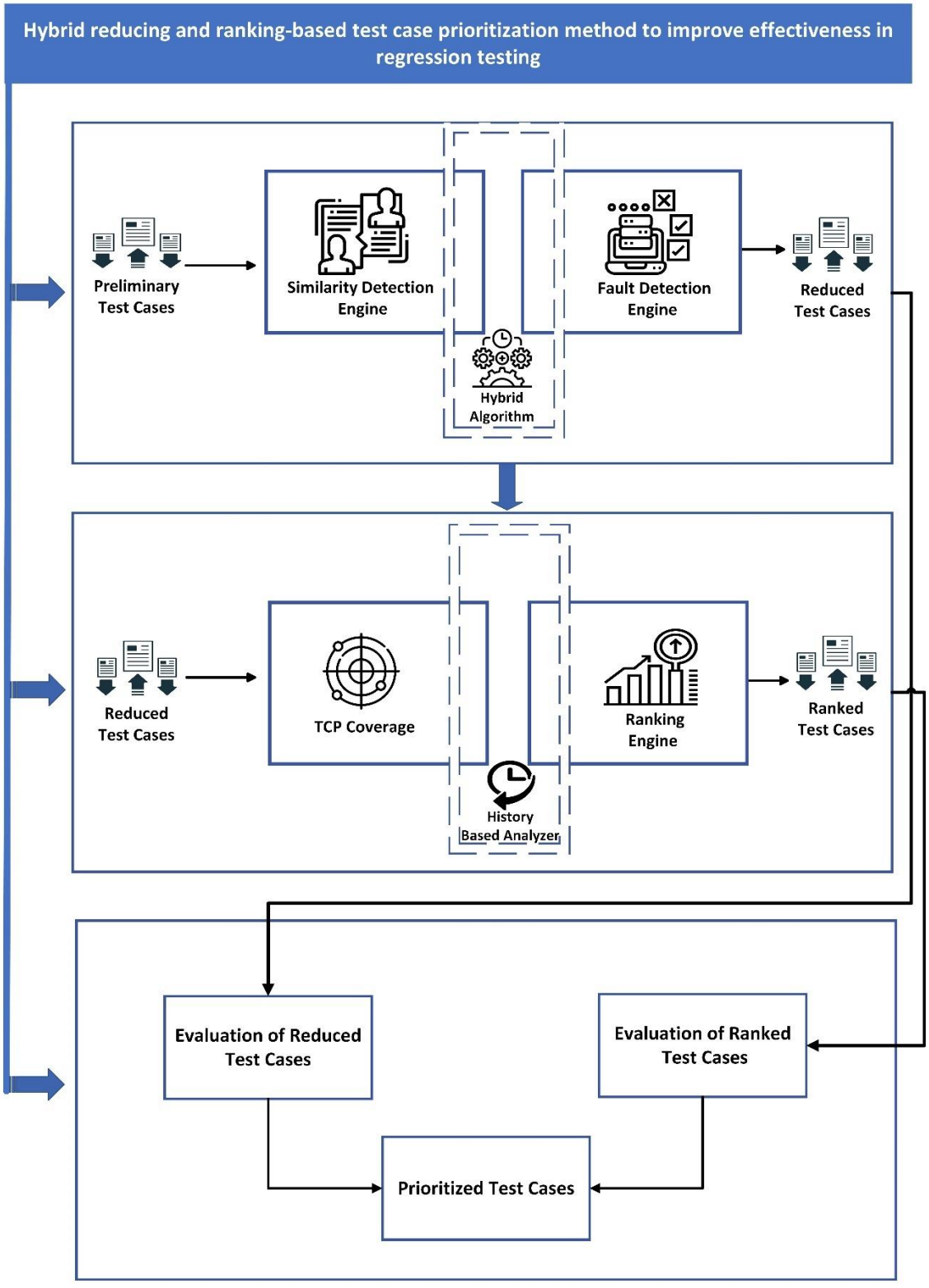

Fig. 2. Hybrid framework

6. Proposed Hybrid Framework

Initially the experiment setup will not involve many test cases, but after the experiment is hybridized, the experiment requires two test plans (TP) for merging and prioritizing. This process will initially execute once per each test case input as shown in Figure.2. Then, a general Hybrid process is proposed to execute all subsequent releases of test cases created by similarity-based and fault-based approaches, as illustrated in Figure 2. In the initial run, the TCs are retrieved from a large, indexed database of textual TCs. The input to the TC repository is retrieved from the experiment dataset, which in this case is the software infrastructure repository (SIR). Each TCs has one input string, which are ordered to run the

system. The dataset for the secondary experiment comes with test inputs and fault matrix. Those outputs are used entirely in this experiment. The test inputs will undergo a similarity process, while the fault matrix will undergo fault process. The generated test plan will then be merged and ordered according to a combination of relevance criteria.

The weighted scoring is measured according to the scale set by the test engineer. The process begins with prioritization and ends with selection. The hybrid process includes three main processes:

I. Test Plan creation based on similarity with prioritization technique.

II. Test Plan creation based on the fault with prioritization technique.

III. Hybridize and apply a selection of the Test Plans created by two previous processes using the weight-based approach.

The final Test Plan will be executed and evaluated to prove that the generated test plan can deliver a better regression result. All execution or processes are run in a controlled experiment setup, developed specifically for this research using JavaScript.

7. Preliminary Results

In this preliminary study four iteration of test suits were used for experimentation 5,10,15 and 20 test cases were used for examining ranking, fault reduction, time and cost of test cases, cluster was formed first to facilitate the process of ranking. After the formation of cluster groups, the ranking of the test cases in the two developed cluster groups was much easier and efficient.

8. Conclusion

This study examines TCP and TCS hybrid testing methods that shorten test runs, lower costs, and increase the effectiveness of regression testing. Regression testing has established apparent norms, advantages, and limitations for each TCS and TCP technique. The State-of-the-Art Hybrid Approaches review offers an alternative testing solution to the industry, helping it choose which testing method is most effective and efficient given the testing assignment schedule and the resources at hand. The main goal of regression test case selection and prioritization is to select only the modified test cases that determine quality parameters such as time, cost and efficiency are important to improve regression testing techniques. The proposed hybrid test case selection technique will exclude similar & faulty test cases to reduce test size. The proposed hybrid technique has several advantages, including reduced execution time and improved fault detection ability. The first step in the research was to observe the pattern of recent publications of related papers in this field of study. The comparative analysis of the research under consideration reveals that the hybrid TCP and TCS methodologies being utilised in regression testing are on the rise and are more effective. The review reveals that the improvements of earlier research on the TCP and TCS approaches have elevated to the forefront of the conversation. This study examines which combinations of regression testing (TCP, TCM, and TCS) approaches are appropriate for use as hybrid procedures that could increase the efficacy and efficiency of subsequent research. This study concludes that the TCP and TCS hybrid technique is more effective for regression testing in terms of time and expense.

References

[1] Fraser, G. and J.M. Rojas, Software Testing, in Handbook of Software Engineering, S. Cha, R.N. Taylor, and K. Kang, Editors. Springer International Publishing: Cham. (2019), pp. 123-192.

[2] Celik, A., et al. Regression test selection across JVM boundaries. in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. (2017).

[3] Singhal, S., et al., Systematic Literature Review on Test Case Selection and Prioritization: A Tertiary Study. Applied Sciences, Vol. 11, No. 24, (2021), p. 12121.

[4] Luo, Q., K. Moran, and D. Poshyvanyk, A large-scale empirical comparison of static and dynamic test case prioritization techniques, in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Association for Computing Machinery: Seattle, WA, USA. (2016), pp. 559-570.

[5] Harrold, M.J. Testing: a roadmap. in Proceedings of the Conference on the Future of Software Engineering. (2000).

[6] Agarwal, E., A review of test prioritization regression testing based on time. Globus An International Journal of

Management and IT, 2019. Vol. 11, No. 1, (2019), pp. 35-38.

[7] Chen, Y., R.L. Probert, and D.P. Sims. Specification-based regression test selection with risk analysis. in CASCON. (2002).

[8] Singh, A., R. Bhatia, and A. Singhrova, Object Oriented Coupling based Test Case Prioritization. International Journal of Computer Sciences and Engineering, Vol. 6, (2018), pp. 747-754.

[9] Banias, O., Test case selection-prioritization approach based on memoization dynamic programming algorithm. Information and Software Technology, Vol. 115, (2019), p. 119-130.

[10] Rahmani, A., J.L. Min, and A. Maspupah, An empirical study of regression testing techniques. Journal of Physics: Conference Series, Vol. 1869, No, 1, (2021), p. 012080.

[11] Vescan, A., C.-M. Pintea, and P.C. Pop, Test Case Prioritization-ANT Algorithm With Faults Severity. Logic Journal of the IGPL, Vol. 30, No. 2, (2022), pp. 277-288.

[12] Bajaj, A. and O.P. Sangwan, A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms. IEEE Access, Vol. 7, (2019), pp. 126355-126375.

[13] Kumar, A. and K. Singh, A Literature Survey on test case prioritization. Compusoft, Vol. 3, No. 5, (2014), pp. 793-799.

[14] Ghani, I., et al., Improved Test Case Selection Algorithm to Reduce Time in Regression Testing. Computers, Materials \& Continua, Vol. 72, No. 1, (2022), pp. 635-650.

[15] Agrawal, A., A. Choudhary, and A. Kaur, An Effective Regression Test Case Selection Using Hybrid Whale Optimization Algorithm. International Journal of Distributed Systems and Technologies, Vol. 11, (2020), pp. 53-67.

[16] Agrawal, A.P., A. Choudhary, and A. Kaur, An effective regression test case selection using hybrid whale optimization algorithm. International Journal of Distributed Systems and Technologies (IJDST), Vol. 11, No. 1, (2020), pp. 53-67.

[17] Zhang, L. Hybrid regression test selection. in 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). (2018).

[18] Haider, A.A., A. Nadeem, and S. Akram, Safe regression test suite optimization: A review. 2016 International Conference on Open Source Systems & Technologies (ICOSST), (2016), pp. 7-12.

[19] Khatibsyarbini, M., et al., Test case prioritization approaches in regression testing: A systematic literature review. Information and Software Technology, Vol. 93, (2018), pp. 74-93.

[20] Magalhães, C., et al., HSP: A hybrid selection and prioritisation of regression test cases based on information retrieval and code coverage applied on an industrial case study. Journal of Systems and Software, Vol. 159, (2020), p. 110430.

[21] Ali, S., et al., Enhanced regression testing technique for agile software development and continuous integration strategies. Software Quality Journal, Vol. 28, (2020).

[22] Panda, S. and D.P. Mohapatra, Regression test suite minimization using integer linear programming model. Software: Practice and Experience, Vol. 47, (2017), pp. 1539-1560.

[23] Arrieta, A., et al., Multi-objective black-box test case selection for cost-effectively testing simulation models. (2018), pp. 1411-1418.

[24] Kazmi, R., et al., A Test Case Selection Framework and Technique: Weighted Average Scoring Method. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), Vol. 9,

Nos. 3-4, (2017), pp. 15-22.

[25] Jahan, H., Z. Feng, and S.M.H. Mahmud, Risk-Based Test Case Prioritization by Correlating System Methods and Their Associated Risks. Arabian Journal for Science and Engineering, Vol. 45, (2020).

[26] Bach, T., R. Pannemans, and S. Schwedes. Effects of an Economic Approach for Test Case Selection and Reduction for a Large Industrial Project. in 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), (2018).

[27] Engström, E., P. Runeson, and A. Ljung. Improving Regression Testing Transparency and Efficiency with History-Based Prioritization -- An Industrial Case Study. in 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation. (2011).

[28] Malhotra, R., A. Kaur, and Y. Singh, A Regression Test Selection and Prioritization Technique. Journal of Information Processing Systems, Vol. 6, No. 2, (2010), pp. 235-252.

[29] Nagar, R., et al., Implementing test case selection and reduction techniques using meta-heuristics. (2014), pp. 837-842.

[30] Nardo, D., et al., Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system. Software Testing, Verification and Reliability, Vol. 25, (2015).

[31] Sampath, S., R. Bryce, and A. Memon, A Uniform Representation of Hybrid Criteria for Regression Testing. Software Engineering, IEEE Transactions on, Vol. 39, (2013), pp. 1326-1344.

[32] Suri, B. and S. Singhal, Analyzing test case selection & prioritization using ACO. ACM SIGSOFT Software Engineering Notes, Vol. 36, (2011), pp. 1-5.

[33] Tyagi, M. and S. Malhotra, Test case prioritization using multi objective particle swarm optimizer. (2014), pp. 390-395.

[34] Saber, T., et al. A Hybrid Algorithm for Multi-Objective Test Case Selection. in 2018 IEEE Congress on Evolutionary Computation (CEC). (2018).

[35] Kandil, P., S. Moussa, and N. Badr, Cluster-based Test Cases Prioritization and Selection Technique for Agile Regression Testing. Journal of Software: Evolution and Process, Vol. 29, (2016).

[36] Spieker, H., et al., Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration. (2017).

[37] Bajaj, A. and O.P. Sangwan, Tri-level regression testing using nature-inspired algorithms. Innovations in Systems and Software Engineering, Vol. 17, No. 1, (2021), pp. 1-16.

[38] Fang, C., et al., Similarity-based test case prioritization using ordered sequences of program entities. Software Quality Journal, Vol. 22, (2014).

[39] Yan, S., et al., A Dynamic Test Cluster Sampling Strategy by Leveraging Execution Spectra Information. (2010), pp. 147-154.

[40] Zhang, C., et al., An Improved Regression Test Selection Technique by Clustering Execution Profiles. (2010), pp. 171-179.

[41] Kazmi, R., et al., Effective Regression Test Case Selection: A Systematic Literature Review. ACM Computing Surveys, Vol. 50, (2017), pp. 1-32.

[42] Shi, A., et al., Comparing and combining test-suite reduction and regression test selection. (2015), pp. 237-247.

[43] Bajaj, A. and O.P. Sangwan, Discrete and combinatorial gravitational search algorithms for test case prioritization and minimization. International Journal of Information Technology, Vol. 13, No. 2,

(2021), pp. 817-823.

[44] Su, W., et al. A Meta-heuristic Test Case Prioritization Method Based on Hybrid Model. in 2020 International Conference on Computer Engineering and Application (ICCEA). (2020).

[45] Alves, E.L.G., et al. A refactoring-based approach for test case selection and prioritization. in 2013 8th International Workshop on Automation of Software Test (AST). (2013).

[46] Wang, X. and H. Zeng, History-based dynamic test case prioritization for requirement properties in regression testing.(2016), pp. 41-47.

[47] Ammar, A., et al., The Effectiveness of an Enhanced Weighted Method with a Unique Priority Value for Test Case Prioritization in Regression Testing. (2018).

[48] Miranda, B., et al. FAST Approaches to Scalable Similarity-Based Test Case Prioritization. in 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). (2018).

[49] Gao, D., X. Guo, and L. Zhao. Test case prioritization for regression testing based on ant colony optimization. in 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS). (2015).

[50] Lima, J.A.P. and S.R. Vergilio, A Multi-Armed Bandit Approach for Test Case Prioritization in Continuous Integration Environments. IEEE Transactions on Software Engineering, Vol. 48, No. 2, (2022), pp. 453-465.

[51] Chen, J., et al., Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering. Journal of Systems and Software, Vol. 135, (2018), pp. 107-125.

[52] Hyunsook, D. and G. Rothermel, On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques. IEEE Transactions on Software Engineering, Vol. 32, No. 9, (2006), pp. 733-752.

[53] Mor, A., Evaluate the effectiveness of test suite prioritization techniques using APFD metric. IOSR Journal of Computer, Vol. 16, No. 4, (2014), pp. 47-51.

[54] Malishevsky, A.G., et al., Cost-cognizant test case prioritization. (2006), Citeseer.

Follow this article at the following site:

Muhammad Asim Siddique[1*], Wan M.N Wan-Kadir[2] & Johanna Ahmad. The State-of-the-art Hybrid Approaches in Regression Testing. IJIEPR 2023; 34 (4) :1-14
URL: http://ijiepr.iust.ac.ir/article-1-1835-en.html