Computers, Materials & Continua
DOI: 10.32604/cmc.2022.025027
Article

Tech Science Press

Improved Test Case Selection Algorithm to Reduce Time in Regression Testing

Israr Ghani*, Wan M. N. Wan-Kadir, Adila Firdaus Arbain and Noraini Ibrahim

School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM), Johor Bahru, 81310, Malaysia
*Corresponding Author: Israr Ghani. Email: gisrar@graduate.utm.my
Received: 08 November 2021; Accepted: 27 December 2021

Abstract: Regression testing (RT) is an essential but an expensive activity in software development. RT confirms that new faults/errors will not have occurred in the modified program. RT efficiency can be improved through an effective technique of selected only modified test cases that appropriate to the modifications within the given time frame. Earlier, several test case selection approaches have been introduced, but either these techniques were not sufficient according to the requirements of software tester experts or they are ineffective and cannot be used for available test suite specifications and architecture. To address these limitations, we recommend an improved and efficient test case selection (TCS) algorithm for RT. Our proposed technique decreases the execution time and redundancy of the duplicate test cases (TC) and detects only modified changes that appropriate to the modifications in test cases. To reduce execution time for TCS, evaluation results of our proposed approach are established on fault detection, redundancy and already executed test case. Results indicate that proposed technique decreases the inclusive testing time of TCS to execute modified test cases by, on average related to a method of Hybrid Whale Algorithm (HWOA), which is a progressive TCS approach in regression testing for a single product.

Keywords: Test case selection; regression testing; change detection; TCS algorithm; test suite minimization

1 Introduction

There are several vital reasons for projects to use regression testing to make sure that software does not interrupt the running program in the system [1–4]. Regression testing has to arrange with a possibly enormous quantity of variations to reduce time during test case selection. A satisfactory way of RT is to detect and select only the appropriate modified test cases in the running program that are affected by code modifications. The main goal of RTS to reduce test suite size within a given time frame by only selecting the tests that are affected by code modifications. Regression testing needs to be re-executed; all existing test cases are already executed and passed before the modification is completed. However, such an approach unreasonably deserves extra time because a number of test cases would be recurrence each time a modification is completed [3,5,6]. The problematic area for modified parts of a system is called regression test selection (RTS) when extracting a modified subset of modified test cases [3,7–9].

A common regression test case selection procedure for a particular software product is focused on three segments, (A) the Research phase for a retest based on test case modifications, (B) the detection of actual change for modified TC as shown in Fig. 1 the third phase (C) the assembling segment calculation of time selection material from the running test cases collection version. To improve the efficiency, and reduce redundancy, time and unnecessary recurrences as well as fault detected of already executed and passed test cases need depth analysis of modified test suite [10–14]. There should be a focus on mentioned areas as much as possible. Furthermore, there is a need for an enhanced regression testing algorithm to detect altered or modified test cases only to provide an efficient test suite selection approach within given time frame [15–18]. Several approaches were introduced to reduce testing time in regression test case selection [19,20]. Therefore, lack of implementation in medium and large-scale testing environments and some testing approaches are limited either to black box or white box techniques [21–23]. Moreover, existing approaches are also limited either for object-oriented or service-oriented based testing environments. Lack of efficient test case selection in literature also causes the deficiency of testing experiments in introduced approaches throughout selection that are interrelated to the revised portion of System Under Test (SUT) [24–27]. The number of test cases escalation to deal with the modified changes, and at last, it turns out to be substantially challenging to execute every one of them inside limited testing time [28,29]. The highlighted problem also needs to be addressed due to insufficient testing environments reported in several studies [30–32]. To address, above limitations, after assessment and experiments of our proposed approach significantly reduces the average time without missing any fault-revealing to perform regression testing.

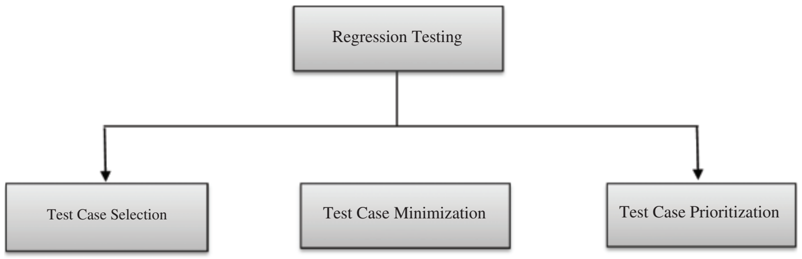

Figure 1: Regression testing

The prominent significances of this study are as follows:

- A new TC) technique using test case selection algorithm to reduce TCS time and redundancy with fault coverage.
- An experiment was performed to verify the time reduction and effectiveness of our approach. Results show the improvements in time reduction by adopting a two-step (test case selection and change detection) procedure without increasing TCS time.

The main goal of this study is to reduce RTCS effort in terms of time (in seconds) and (No. of overall test cases/total no. of modified test cases). We proposed a safe lightweight regression test case selection technique using enhanced test case selection and change detection algorithm by applying mapping rules (test cases change collection history, number of changes calculation, number of modified test cases) for test suites S, SF and TS. The primary purpose of splitting test cases into two different stages is by applying additional filters to detect (exclude Fault Detect test case, duplicate test cases and already executed test case) from the x2 test suite.

2 Related Work

The main purpose of this study is to introduce an enhanced regression testing approach, simplify the test technique used, find the gaps in regression testing issues that remain to explore in depth and address the test case repetition challenges which increase the extraordinary time and precision in test suit selection. Presently, there are several issues need to address but most of them has been discussed in state-of-the-art testing challenges perspectives on "regression test case selection to reduce time and improve precision in software testing area. TCS and detection of modified test cases in regression testing are being adopted to reduce time and improve precision.

Therefore, the lack of implementation on medium and large scale testing environments and some testing approaches are restricted to black or white boxes. Banias [26] introduced a TCS algorithm based on dynamic memorization programming. Therefore, the main focus was the management of a huge amount of test cases with the allocated memory size of the system, which is not close to real project scenarios. Souza et al. [28] recommended a search constructed algorithm, but this technique is limited to functional and object-oriented based applications. Harikarthik et al. [30] developed a TCS algorithm, therefore this approach discusses the test case prioritization and the proposed algorithm was not compared with more than one approach. Delavernhe et al. [32] published a multi-objective test case selection algorithm. Therefore, the proposed algorithm was not compared to another approach to assess the time and effectiveness.

Beleová et al. [33] presented hardware compatibility related test cases; however, minimum and maximum requirements of the application is not mentioned. Additionally, this algorithm is limited to functional testing only. Sahoo and Ray [34] proposed a slicing based framework and algorithm. Their proposed algorithm is limited to select modified test cases for WSDL files only, also proposed algorithm results have not been compared to know the efficiency of the proposed framework and algorithm. In 2018, Wang et al. [25] introduced RTS approach, but their approach was not compared to relevant regression-based test case selection approaches. Chen and Zhang [35] proposed a predictive based test case selection approach where multiple iterations have been performed on each test suite, but results are now shown for each iteration, and the focus was only on fault detection. In 2019 Wang et al. [25] also introduced the RTS approach, but their approach was not compared to relevant regression-based test case selection approaches. Earlier, authors have introduced RTS techniques using BAT, HWOA and ACO by allowing for medium or large size test suite so the total time of TCS can be evaluate and reduced efficiently [25,36–38]. Therefore, these approaches are limited either for objected based applications or service-oriented based test cases where multiple hardcode checks have also been performed to get expected results. To address above mentioned weaknesses and limitations, we propose an improved and efficient test case selection algorithm for regression testing. We believe that the enhanced test case selection and change detection algorithm of regression testing will resolve highlighted research problems; an enhanced regression testing approach will improve testing quality. An enhanced regression testing approach will reduce the time concerns and questions raised in this research despite the facts. Regression testing analysis and experiments show how the regression test case selection workflow has been adopted to address the mentioned concerns and questions that have been proposed in our research.

3 Problem Statement

RTS is endless and immeasurable activity to execute the complete test suite. Due to this reason, need to identify the modified test case selection by using change detection algorithm on the test case selection procedure between modified and actual test suits.

In next step, a dynamic algorithm requires to select only the modified and the affected parts of the requested change for time, cost and efficiency in regression testing. Furthermore, need to improve to avoid repetition in test case selection and an efficient regression testing approach to select an efficient test case suite. RTS is of major concern and described as follow:

Given: A test suite S = {s1, s2, s3, . . . , sn}, a set of fault and redundant test suite SF = {sf1, sf2, sf3, . . . , sn}, selected by change detection criterion C and final subset of TS {ts1, ts2, ts3, . . . , Tn}, that are detected by each of the SF such that each modified test case TS (TS∈S) which covers fault of S and SF. With total execution time Δtimeh.

Regression testing is vital when changes have been introduced during software enhancement. Therefore, the more significant challenge is selecting an efficient test suite within the given time frame. In order to select modified test cases and detect only modified changes, Fig. 2 explains and divides one test suite into three steps. Suite_1 is a combination of original and modified test cases. Suite_2 detects the modified test case affected by code changes, and Suite_3 provides an actual test suite to execute, which contains only modified test cases. In order to reduce time in test case selection and detection of modified change, we split the initial test suite into mentioned three steps.

$$\text{RTS} : S > SF \tag{1}$$

$$\text{Total execution time } \Delta = SF : SF > TS \tag{2}$$

$$TS : TS < SF < S \tag{3}$$

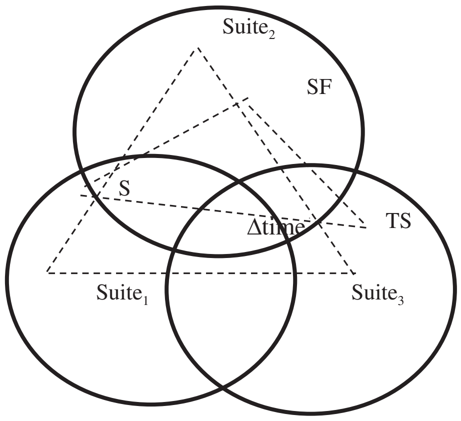

Figure 2: Problem statement of TCS

4 Proposed Solution

In this study, an enhanced RTCS using test case selection algorithm (modified change detection) approach is proposed, enhancement direction proposed to reduce time and improve precision. The tackling of the persistent uncertainty in parameter selection and determination of appropriate regression testing approach. Preliminary findings and are presented with appropriate examples and the proposed framework and algorithm is included in this study.

To define the effectiveness and outcome of the proposed algorithm, we introduced a framework and algorithm with the following three steps. Therefore, to reduce three different test suites size S, SF and TS, a total of four iterations have been performed for each test suite as 50, 100, 150, and 200 test cases compared to propose test cases by Hybrid Whale Optimization Algorithm (HWOA) then compare its ratio in terms of time. The main purpose is to define the workflow to get the statistical evidence of the proposed solution compared to the HWOA algorithm.

4.1 Experimental Framework

The following test case selection and change detection framework are well-structured in a planned manner to assist the researchers in accomplishing the research goals. Fig. 3 shows the research framework according to the study's objectives described in the experimental setup section for an enhanced RT approach to select the modified test cases and detect the newly added change that is proposed and raised the questions in this research. The stepwise phases are described below and depicted as followed:

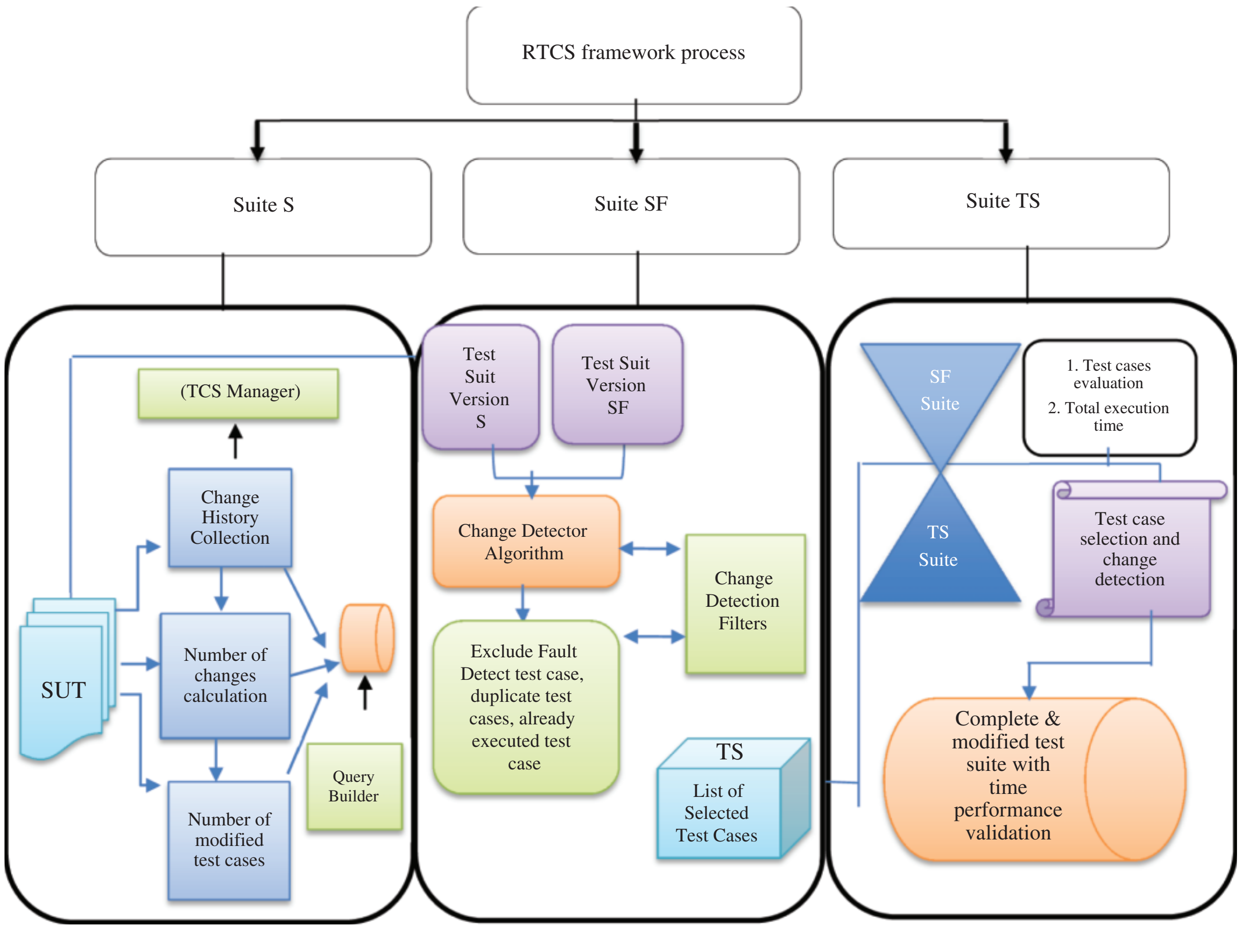
RTCS framework process
Suite S
Suite SF
Suite TS
(TCS Manager)
Change History Collection
SUT
Number of changes calculation
Number of modified test cases
Query Builder
Test Suit Version S
Test Suit Version SF
Change Detector Algorithm
Change Detection Filters
Exclude Fault Detect test case, duplicate test cases, already executed test case
TS
List of Selected Test Cases
SF Suite
TS Suite
1. Test cases evaluation
2. Total execution time
Test case selection and change detection
Complete & modified test suite with time performance validation

Figure 3: RTCS framework

This analysis will have examined the structure of the efficient test case selection and detection of newly added change from test suite X while considering the role to read the activity from SUT. Each word plays in the sentences where change collection history, number of modified changes calculation and total number of modified test cases against SUT (System Under Test) to select modified test cases for modified test suite X^1 and X^2. The planning from literature review phase consisted of three main elements: problem formulation, literature review and identifying existing testing approach. Problem formulation involved the process of identifying the issues that existed in regression testing approach that did not have solutions or if a solution was available it needed much improvement. The broad problem addressed in this research is an enhanced regression testing approach in test case selection and detection of modified change to reduce testing time. The problem is narrowed down to test case selection and further narrowed down to the specific issue of detection in modified test case. The RTCS framework is used on a medium and large scale database to detect modified test cases and significant improvement compared to previous application on test suit selection and detection of modified test cases in regression testing.

The framework should be well-structured in a planned manner to assist the researchers in accomplishing the research goals according to the given workflow. Fig. 3 shows the research framework according to the TCS procedure raised in this study described in Fig. 1. To select the enhanced RT technique for selection of modified test cases and detect the newly added change proposed in this research. This analysis has examined the efficient test case selection structure and detection of newly added change from test suite S. The primary purpose is to split one test suite into two stages to perform efficient test case selection and detect a modified change from modified test cases. The first stage refers to modified test cases only, while the second stage detects newly added changes in the modified test case. We claim that using two different iterations for one test suite will provide an efficient test suite to execute.

However, considering the role of reading the activity from SUT where change collection history, number of modified changes calculation, and the total number of modified test cases against SUT (System under Test) to select modified test cases produce test suite SF. The second part of the framework shows the detection of newly added change in modified test cases where a change detection algorithm is applied for small, medium and large scale test suites. Multiple filtered applied to exclude duplicate, irrelevant, fault detector and already executed test cases according to the requirements to get the updated and efficient version of modified test suite TS using the query builder. The Stanford rich text format based library and WSDL files have been used for this purpose. It also derives the relationships among the different components. Following is the process flow of the proposed framework.

4.2 Implementation of RTCS Algorithm

The first stage of TCS is the selection procurement; in this stage, the test case selection and change detection algorithm is used to select the modified change. The second stage has been introduced for change detection to detect the newly added change in the modified part of the test case and evaluate the proposed algorithm RTCS with quality parameter time. This segment describes the process flow and procedures followed to achieve the objectives of the study. The process flow also defines the detailed design of the conducted research which leads to the development of an enhanced testing approach to diagnose issues from test case selection in regression testing, extraction of test case cases based on regression test case selection process flow. The ultimate aim of any enhanced test case selection and change detection testing approach is to get the needed pure vein pattern and describe its features.

The test case selection and change detection process flow in Fig. 4 is presented and then applies a dimensionality reduction on test case repetition features using the change detection filters. Furthermore, an application using an enhanced regression testing approach supports selecting an efficient test suit technique. Based on test cases, selection and change detection algorithms are applied to avoid over-fitting, yield a better generalization test suite, and improve the regression test case technique. The novelty of this study relies on the test case selection and detection of modified change in regression testing for from where the test case selection extracted features are implemented based on the test suite selection.

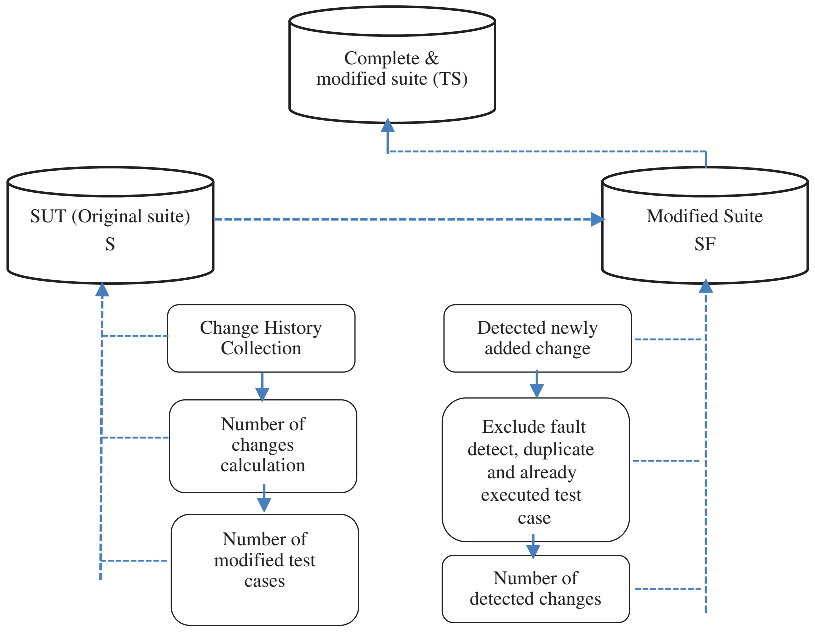
Complete & modified suite (TS)
SUT (Original suite) S
Modified Suite SF
Change History Collection
Number of changes calculation
Number of modified test cases
Detected newly added change
Exclude fault detect, duplicate and already executed test case
Number of detected changes

Figure 4: RTCS process flow

4.3 Test Case Selection Algorithm

In the process of TCS if the selection of test cases is stated as minor (S) level (50 test cases), which fault reports, so this execution can be negated for the upcoming iteration. In detecting actual modified test cases (SF) segment, duplicate, false and already executed test cases will test to get expected results within the given time frame. The mentioned selection and detection procedure has been performed to target only invalid test cases. To insure about mentioned criteria, the test case selection algorithm is introduced and shown as follows:

```
Algorithm:        SelectTestCase (S, S(F), ValidInput. ValidOutput) : S(F)'
Input:            S. S' the original/enhanced test suite,
S(F) A false test suite created to test S
Output: S(F)'- Subset of S(S') selected for executing S(F)
          begin
                S(F)' =Ø
                           i = 1
B = Build SF from S
d = Select false test cases from S(F)
while NOT end of log file (S(F))
          begin
```

(Continued)

Algorithm: Continued

```
                select $d_i$
case S1 (ValidInput = FALSE) and (ValidOutput = TRUE):
       begin
              $B_{si}$ = build SF from S
              If InvokeProcedureCall (S, $S_{si}$)a
                     break;
              else if NonRedundantTestCase (di, S(F)’ b)
                            break:
                     else S(F)’ = S(F)’ + di
              end
case S2 (ValidInput = FALSE) and (ValidOutput = TRUE):
       begin
               if NonRedundantTestCase ($d_i$, S(F), S(F)’b)
                   break;
       else S(FY = S(F) + $d_i$,
       end
case S3 (ValidInput = FALSE) and (ValidOutput = TRUE):
             begin
                   d(S) = d(S)’ + $d_i$
             End
             i = i + 1
             end
      Return S(S)’
end
```

Although a ValidInput is recognized will be justified for the upcoming test suite and will remain the same as the part of the major (SF) original and modified suite. Therefore a false and enhanced test suite has been generated to check the redundancy for the next iteration. However, this procedure will continue unrestrictedly for every fault test case until the method completes its detection finally.

The process of ValidInput and ValidOutput are non-compulsory of the recommended method which demanded to be input. Furthermore, the original input of the test suite (SF) and modified input (S’) will be responsible for result of false test cases within the given time frame and test suite.

Overall, four iterations have been performed for 50, 100, 150, 200 test cases by followed the process mentioned in Figs. 3, 4. We can see in Figs. 5 to 8 how efficiently time has been reduced by using RTCS algorithm, where results are well described in Tab. 1 for further analysis.

Tab. 2 shows the average calculation time for the TC selection phase as per Fig. 3 framework and compared with the HWOA approach using average ratio time.

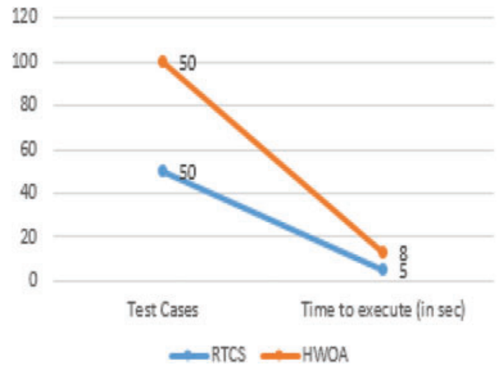

Figure 5: 50 Test cases

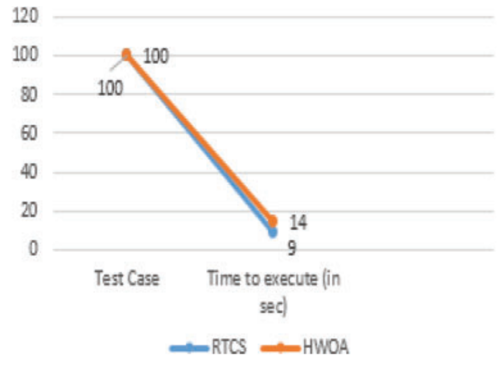

Figure 6: 100 Test cases

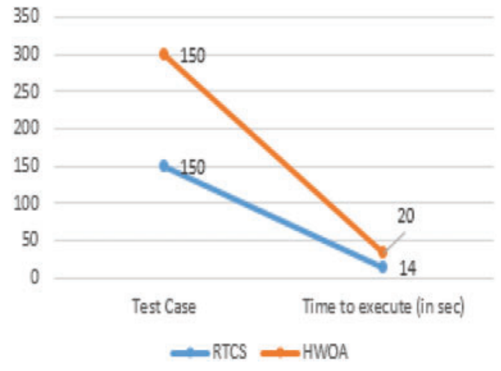

Figure 7: 150 Test cases

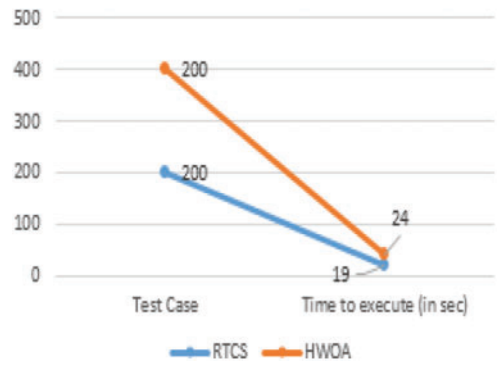

Figure 8: 200 Test cases

Table 1: Stepwise execution plan

Input	Test case selection Data from the selected database, API and interface
Output	An efficient test suite to execute after selection and detection modified test cases
Begin	
	Step 1: Test case selection using test case selection mapping rules
	Step 2: Divide Test Suite Selection and detection test suites
	Step 3: Insert test cases into the selected database to select modified test cases
	Step 4: Apply change detection algorithm against selected modified test cases to exclude duplicate/fault detector and already executed test cases from the modified test suite.
	Step 5: send the request to detect the modified test cases.
	Step 6: Receive a response for the selected and modified changes from the test suite
	Step 7: calculate time against of selected and detected unique test cases from modified test suite.
	Step 8: perform analysis to compare with HWOA approach on the expected result
End	

Table 2: Test case selection time

Approaches	Test cases			
	50 TC	100 TC	150 TC	200 TC
RTCS (avg time)	5	9	14	19
HWOA (avg time)	8	14	20	24

4.4 Change Detection Algorithm

Meanwhile the modified test suite (SF) produced from the recommended approach after excluding redundant fault detectors and already executed test cases from (S). (SF) is a set of false and redundant test cases that need to be associated with false test case in (TS) with the same test cases numbers as compare to HWOA. A proposed procedure named the "ChangeDetectorTestCase" algorithm is as follows:

```
Algorithm:       NonRedundantTestCase (TS): TRUE or FALSE
Input ti-        Mark false test case,
Input $\Delta time_h$-    Calculate Execution time,
                 S(F)'- Subset of (S) selected for executing $\Delta time_h$ '
Output:          TRUE when t is not redundant pathway, ELSE proceeds FALSE
          begin
                 $k = 1$
                 $C_{ti}$ = Generate SF from $S_i$
                 $t_j$ = Detect false test cases k from SF'
                 $C_j$ = Construct TS from $S_j$
                 while (NOT end of (SF) and ($C_{ti}$ <> $C_j$)
          begin
                 $j = j + 1$
                 $\Delta time_{h=}\ \Delta time_h + 1$
                 $S_j$ = Detect a false test j from (SF)'
                 C = Construct TS from SF
          End
                 if ($C_{ti}$ == ($C_j$,)
                       Return TRUE
                       Return $\Delta time_h$
          Else Return FALSE
end
```

Lastly, the result will be a decreased test suit (TS) size within given time frame. Following are the results for the execution of the mentioned algorithm in four iterations. It shows how our technique reduces the time by splitting test case selection in the first phase compared to the HWOA technique. Results indicates its benefits in terms of detecting case and false test suite SF can be squeezed to generate an optimized test suite. Finally, the modified (TS) will be produced after applying multiple filters to exclude duplicate test cases to redundancy and cases which are already executed in test suite (SF). Finally, TS with given time Return Δtime and the results are presented as follows:

Maximum four iterations have been performed for 50, 100, 150, 200 test cases for change detection by followed the process mentioned in Figs. 3, 4. We can see in Figs. 9 to 12 how proficiently time has been reduced using a change detection algorithm where results are well described in Tab. 2 for further analysis.

Tab. 3 explains the average calculation time for change detection phase for modified test cases as per Fig. 3 framework and well compared with HWOA approach using average ratio time. The test case selection and change detection algorithm can tackle the low, medium and high volume number of test cases.

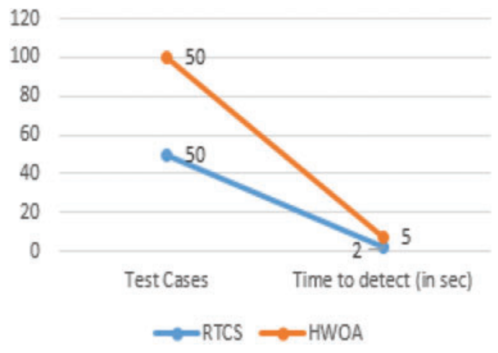

Figure 9: 50 Test cases

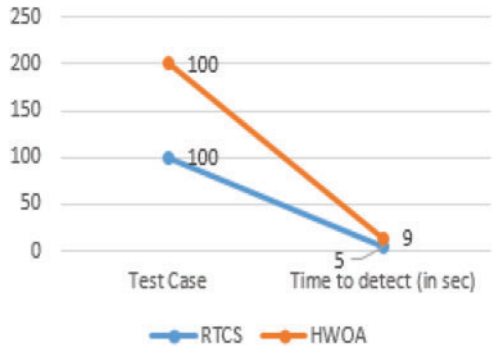

Figure 10: 100 Test cases

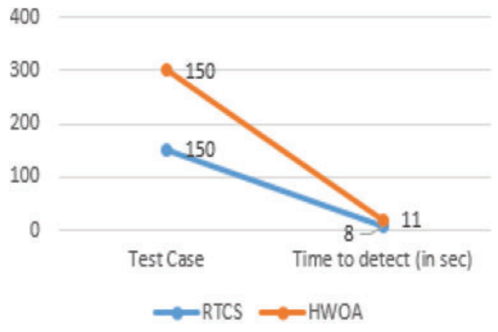

Figure 11: 50 Test cases

Our approach have the ability to select only modified and approved test cases repetition problems in regression testing as well as this approach able to have an enhanced regression test testing to reduce time and improve precision within given timeframe.

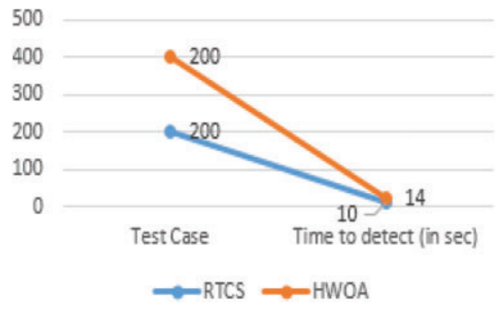

Figure 12: 50 Test cases

Table 3: Test case selection time

Approaches	Test cases			
	50 TC	100 TC	150 TC	200 TC
RTCS (avg time)	2	5	8	10
HWOA (avg time)	5	9	11	14

5 Threats to Validity

The purpose of regression testing is to ensure that software satisfies the requested customer requirements [21,39,40]. In this section, some of the possible threats to validity of results of this study have been discussed. Corrupted data or incorrect program execution in ASP.NET, SQL server management and API's test cases are actual threats in internal validity in this research. The experimental procedure is implemented in ASP.NET 3.5 framework installed on a pc with Intel @ Court i7-3770 CPU @ 3.40 GHz 3.40 GHz installed with 16 GB ram. The conclusion of this research is consistent but there is a possibility of validity and bias threats for the regressing based blackbox testing against selected test cases to create test suite of selection, conduct analysis, experimental setup process or result phase of the study. Therefore, in regression testing has some fundamental concerns like effectiveness and accuracy in test case selection and change detection with consistency. The proposed algorithm to recognize the primary study is concluded by deliberating bias, external validity and internal validity suggested and suggested by Berk et al. [41]. For further validation, the unambiguousness of the proposed study will contribute to evaluating the reliability of the effects results for the researchers. The regression test case selection and change detection results determine that this study's justification is vital and needs more attention, as discussed in the introduction section. In the foregoing, an enhanced regression test case selection and change detection approach need more concentration for integrity, security, interoperability and inconsistencies of regression testing [42].

There is the probability to avoid some studies for the reason that of grey areas of study related to regression test case selection, such as the contribution of solution techniques and the relationship between test case selection and change detection, testing techniques and testing levels. This study is connected with numerous software testing group of people, quality assurance, information systems, service oriented architecture (SOA), and web service testing. An enhanced regression testing approach to select test suite persistent uncertainty in quality parameters selection and detection to determine appropriate and sensitive quality attributes such as time and efficiency are essential, which put the business of the software industry at stake.

6 Conclusion

The main goal of regression test case selection and detection of the modified change approach is to select a test suit that determines quality parameters such as time and efficiency are important to improve regression testing techniques. The result evaluation shows that the analysis efficiently identifies how by splitting the test case selection and detecting modified changes to reduce TCS time in three phases. (A) The analysis or exploration phase selects TC for a retest based on test case modifications, (B) the detection of actual change for modified test cases and (C) the collection phase calculation of the time of selection material from the running test suite version. Our proposed approach significantly reduces TCS execution time by following the mentioned three phases to getting expected results and improves the TCS technique within the given time frame. Our proposed approach decreases the execution time and redundancy of the duplicate TC and detects only modified changes. Therefore, selecting test suite persistent uncertainty in quality parameters selection and detection to determine appropriate and sensitive quality attributes such as efficiency and accuracy is also important, which puts the software industry's business at stake. For future research, an enhanced regression testing approach needs more concentration for performance, accuracy, security, and re-usability of regression testing.

Funding Statement: This work was supported in part by the Research Management Center (RMC), Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education Malaysia (MOHE) through the UTM High Impact Research (UTMHR) grant scheme under (Vot Number Q.J130000.2451.08G55).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. Garousi, R. Özkan and A. Betin-Can, "Multi-objective regression test selection in practice: An empirical study in the defense software industry," *Information and Software Technology*, vol. 103, pp. 40–54, 2018.
[2] R. Kazmi, D. Jawawi, R. Mohamad and I. Ghani, "Effective regression test case selection: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–32, 2017.
[3] S. Khan, S. Lee, N. Javaid and W. Abdul, "A systematic review on test suite reduction: Approaches, experiment's quality evaluation, and guidelines," *IEEE Access*, vol. 6, pp. 11816–11841, 2018.
[4] B. Guan, Y. Zhao and Y. Li, "An improved ant colony optimization with an automatic updating mechanism for constraint satisfaction problems," *Expert Systems with Applications*, vol. 164, pp. 114021, 2021.
[5] W. Lewis, "Prepare for the next spiral (Act)," in *Software Testing and Continuous Quality Improvement*, Auerbach Publications, New York, USA, pp. 251–260, 2017.
[6] C. Zhu, O. Legunsen, A. Shi and M. Gligoric, "A framework for checking regression test selection tools," in *IEEE/ACM 41st Int. Conf. on Software Engineering (ICSE)*, Montreal, QC, Canada, IEEE, pp. 430–441, 2019.
[7] D. Pradhan, S. Wang, S. Ali and T. Yue, "Search-based cost-effective test case selection within a time budget: An empirical study," in *Proc. of the Genetic and Evolutionary Computation Conf.*, Boston, USA, vol. 2016, pp. 1085–1092, 2016.
[8] W. Linden and H. Ren, "A fast and simple algorithm for Bayesian adaptive testing," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 1, pp. 58–85, 2020.

[9] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: A survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[10] D. Pradhan, S. Wang, S. Ali, T. Yue and M. Liaaen, "CBGA-ES: A cluster-based genetic algorithm with elitist selection for supporting multi-objective test optimization," in *IEEE Int. Conf. on Software Testing, Verification and Validation (ICST)*, Tokyo, Japan, IEEE, pp. 367–378, 2017.

[11] A. Ngah, M. Munro, Z. Abdullah, A. Jalil and M. Abdallah, "Regression test selection model: A comparison between ReTSE and pythia," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 2, pp. 844–851, 2019.

[12] A. Agrawal, 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, pp. 53–67, 2020.

[13] P. Mishra and L. Singh, "Test case selection for regression testing of applications using web services based on wsdl specification changes," in *Int. Conf. on Computing, Communication & Automation*, Greater Noida, India, IEEE, pp. 908–913, 2015.

[14] A. Mustafa, W. Kadir and I. Ibrahim, "Comparative evaluation of the state-of-art requirements-based test case generation approaches," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 7, no. 4–2, pp. 1567–1573, 2017.

[15] O. Legunsen, F. Hariri, A. Shi and Y. Lu, "An extensive study of static regression test selection in modern software evolution," in *Proc. of the 24th ACM SIGSOFT Int. Symp. on Foundations of Software Engineering*, Auckland, Newzealand, pp. 583–594, 2016.

[16] A. Choudhary, A. Prakash and A. Kaur, "An effective approach for regression test case selection using pareto based multi-objective harmony search," in *Proc. of the 11th Int. Workshop on Search-Based Software Testing*, New York, USA, pp. 13–20, 2018.

[17] G. Guizzo, J. Petke, F. Sarro and M. Harman, "Enhancing genetic improvement of software with regression test selection," in *IEEE/ACM 43rd Int. Conf. on Software Engineering (ICSE)*, Madrid, Spain, IEEE, pp. 1323–1333, 2021.

[18] A. Mustafa, W. Kadir and N. Ibrahim, "Automated test case generation from requirements: A systematic literature review," *Computers Materials & Continua*, vol. 67, no. 2, pp. 1819–1833, 2021.

[19] M. Harman, Y. Jia and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *IEEE 8th Int. Conf. on Software Testing, Verification and Validation (ICST)*, Graz, Austria, IEEE, pp. 1–12, 2015.

[20] M. Refai, W. Cazzola and S. Ghosh, "A fuzzy logic based approach for model-based regression test selection," in *2017 ACM/IEEE 20th Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*, Austin, TX, USA, IEEE, pp. 55–62, 2017.

[21] M. Gligoric, L. Eloussi and D. Marinov, "Practical regression test selection with dynamic file dependencies," in *Proc. of the Int. Symp. on Software Testing and Analysis*, New York, USA, pp. 211–222, 2015.

[22] W. Zheng, R. Hierons, M. Li, X. Liu and V. Vinciotti, "Multi-objective optimisation for regression testing," *Information Sciences*, vol. 334, pp. 1–16, 2016.

[23] M. Khatibsyarbini, M. Isa, D. Jawawi and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," *Information and Software Technology*, vol. 93, pp. 74–93, 2018.

[24] D. Panwar, P. Tomar and V. Singh, "Hybridization of Cuckoo-ACO algorithm for test case prioritization," *Journal of Statistics and Management Systems*, vol. 21, no. 4, pp. 539–546, 2018.

[25] K. Wang, C. Zhu, A. Celik and J. Kim, "Towards refactoring-aware regression test selection," in *IEEE/ACM 40th Int. Conf. on Software Engineering (ICSE)*, Gothenburg, Sweden, IEEE, pp. 233–244, 2018.

[26] O. Banias, "Test case selection-prioritization approach based on memoization dynamic programming algorithm," *Information and Software Technology*, vol. 115, pp. 119–130, 2019.

[27] A. Bertolino, G. Angelis and F. Lonetti, "Governing regression testing in systems of systems," in *IEEE Int. Symp. on Software Reliability Engineering Workshops (ISSREW)*, Berlin, Germany, IEEE, pp. 144–148, 2019.

[28] L. Souza, R. Bastos, C. Prudêncio and F. Barros, "A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection," *Journal of the Brazilian Computer Society*, vol. 21, no. 1, pp. 1–20, 2015.

[29] M. Machalica, A. Samylkin, M. Porth and S. Chandra, "Predictive test selection," in *IEEE/ACM 41st Int. Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, Montreal, Canada, IEEE, pp. 91–100, 2019.

[30] S. Harikarthik, V. Palanisamy and P. Ramanathan, "Optimal test suite selection in regression testing with testcase prioritization using modified ann and Whale optimization algorithm," *Cluster Computing*, vol. 22, no. 5, pp. 11425–11434, 2019.

[31] M. Kargar and A. Hanifizade, "Automation of regression test in microservice architecture," in *4th Int. Conf. on Web Research (ICWR)*, Tehran, Iran, IEEE, pp. 133–137, 2018.

[32] F. Delavernhe, T. Saber, M. Papadakis and A. Ventresque, "A hybrid algorithm for multi-objective test case selection in regression testing," *Congress on Evolutionary Computation*, Rio de Janeiro, Brazil, IEEE, pp. 101–108, 2018.

[33] M. Beleová, Z. Kotásek and T. Hruka, "Application of evolutionary algorithms for regression suites optimization," in *IEEE 18th Int. Symp. on Design and Diagnostics of Electronic Circuits & Systems*, Belgrade, Serbia, IEEE, pp. 91–94, 2015.

[34] S. Sahoo and A. Ray, "A framework for optimization of regression testing of web services using slicing," in *2017 Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, IEEE, pp. 1017–1022, 2017.

[35] L. Chen and L. Zhang, "Speeding up mutation testing via regression test selection: An extensive study," in *2018 IEEE 11th Int. Conf. on Software Testing, Verification and Validation (ICST)*, Gothenburg, Sweden, IEEE, pp. 58–69, 2018.

[36] L. Zhang, "Hybrid regression test selection," in *2018 IEEE/ACM 40th Int. Conf. on Software Engineering (ICSE)*, Gothenburg, Sweden, IEEE, pp. 199–209, 2018.

[37] G. Gay, "The fitness function for the job: Search-based generation of test suites that detect real faults," in *2017 IEEE Int. Conf. on Software Testing, Verification and Validation (ICST)*, Tokyo, Japan, IEEE, pp. 345–355, 2017.

[38] A. Shi, P. Zhao and D. Marinov, "Understanding and improving regression test selection in continuous integration," in *IEEE 30th Int. Symp. on Software Reliability Engineering (ISSRE)*, Berlin, Germany, IEEE, pp. 228–238, 2019.

[39] S. Mansky and E. Gunter, "Safety of a smart classes-used regression test selection algorithm," *Electronic Notes in Theoretical Computer Science*, vol. 351, pp. 51–73, 2020.

[40] S. Khatib, "Optimization of path selection and code-coverage in regression testing using dragonfly algorithm," in *Int. Conf. on Information Technology (ICIT)*, Amman, Jordan, IEEE, pp. 919–923, 2021.

[41] R. Berk, L. Brown, A. Buja, K. Zhang and L. Zhao, "Valid post-selection inference," *The Annals of Statistics*, vol. 41, pp. 802–837, 2013.

[42] M. Jeong and H. Zo, "Preventing insider threats to enhance organizational security: The role of opportunity-reducing techniques," *Telematics and Informatics*, vol. 63, pp. 101670, 2021.