Regression Test Case Prioritization: A Systematic Literature Review

Ali Samad[1], Hairulnizam Mahdin[2], Rafaqut Kazmi[3], Rosziati Ibrahim[4]
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, 86400 Batu Pahat, Johor, Malaysia[1, 2, 4]
Faculty of Computing, The Islamia University of Bahawalpur, 63100 Bahawalpur, Pakistan[1, 3]

***Abstract*—The techniques associated with the Test Case Prioritization (TCP) are used to reduce the cost of regression testing to achieve the objectives that the modifications in the target code would not impact the functionality of updated software. The effectiveness of the TCP is measured based on the cost, the code coverage, and fault detection ability. The regression testing techniques proposed so far are focusing on one or two effectiveness parameters. In this paper, we presented a state-of-art review of the approaches used in regression testing in detail. The second objective is to combine these effective adequacy measures into a single or multi-objective TCP task. This systematic literature review is conducted to identify the state-of-the-art research in regression TCP from 2007 to 2020. The research identifies fifty-two (52) relevant studies that were focusing on these three selection parameters to justify their findings. The results reveal that there were six families of regression TCP in which meta-heuristic regression TCP were reported in 38% and generic regression TCP techniques in 31%. The parameters used as prioritization criteria were cost, code coverage, and fault detection ability. The code coverage is reported by 38%, cost in 17%, and cost and code coverage in 31%. There were three sources for datasets were identified named Software artefact Infrastructure Repository (SIR), Apache Software Foundation, and Git Hub. The measurement and metrics used to validate the effectiveness are inclusiveness, precision, recall, and retest-all.**

Keywords—*Software testing; regression testing; test case prioritization; cost; code coverage; fault detection ability*

I. Introduction

Regression Testing (RT) is an iterative fragment of the software testing and also the primary activity during the maintenance phase. In the literature, it is mentioned that 70% of the testing cost is consumed by regression testing [1]. Once a software system is reorganized, code is modified. Whenever a software needs to be re-tested, the tester may prioritize, select or reduce the test suite size, to achieve multiple objectives of testing like code coverage, fault detection rate, cost of testing, or time. The objective of regression testing is to provide the confidence that changes did not affect the new product and reduce the overall cost of the testing. All these objectives are difficult to achieve in a single testing cycle. If the coverage should increase, cost and time also increased [2]. In regression testing, 100% of code coverage may not be preferred. The efficiency of prioritization may raise the yield of the testing procedure by the means of fault detection ability.

RT helps in testing the code by analyzing the target code both in original and updated form. Furthermore, it performs checking with the assumption that the updates in the target code has minimum or negligible effects on the services provides by the software [3]. The reports claim that code testing is 80% of the total cost of the software cost which is different from the maintenance cost that is about 50% of the total cost [4-6]. One of the objectives of regression testing is to reduce the testing cost by using the state-of-art approaches used in Test Case Selection (RTS), Test Case Prioritization (TCP) along Test Case Reduction (RTR) [7].

The classic techniques of TCP consist of three general components, TCP framework, prioritization parameters and prioritization adequacy measures as shown in Fig. 1. This generalized process takes original program P, modified program P′ and test suites T as input. The prioritization process may have a framework which identifies the code change information from P and P′ and other relevant information like code coverage, fault detection ability, and executional cost. The test case prioritization measure may prioritize the test cases from T and move them to T′ (a subset of T), based on computations performed by selection logic described in the framework. The TCP adequacy measures are used to assess the effectiveness of TCP technique and results produced by this technique.

The TCP adequacy measures are effective in judging the effectiveness of the TCP process. These measures are computed in two ways in the TCP process, the first is to prioritize the test cases on these prioritization contexts like TCP based on coverage measures, TCP based on cost measures, and TCP based on fault rates. The second use is to assess the effectiveness of TCP by coverage or cost optimization. There are four challenges to organizing the three parameters (cost, coverage, and fault detection) in a fashion to assess their importance, dependencies, and priority concerning each other. The other challenge is to choose the appropriate type of these measures, like coverage subtypes, cost subtypes, fault types, and severity. The third challenge is to identify the relevant frameworks that adjust the three parameters for the prioritization of test cases and their adequacy scale and use as adequacy measure. The fourth challenge is to identify the techniques based on these effectiveness measures, such as execution cost, code coverage, and fault detection ability. The primary objective of this paper is to define the TCP effectiveness based on cost, code coverage, and fault detection ability as effectiveness contributors. Furthermore, this survey assesses the current state-of-the-art algorithms in the design of

the regression test case prioritization frameworks and techniques so far. The secondary objectives of this research are to identify the available datasets and methods for the solution of test case prioritization problems.

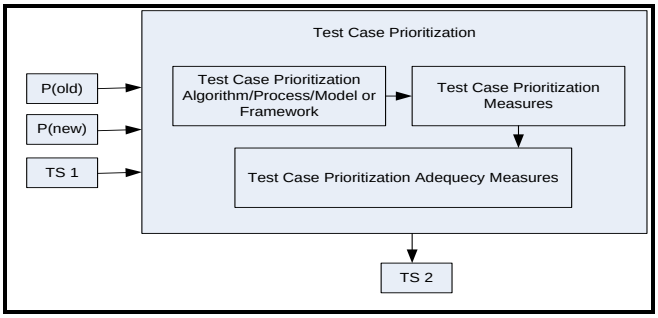

Fig. 1. Maintenance Process Model.

The rest of the paper is organized as follows: Section II formulates the literature selection process of the studies; Section III encompasses data extraction and Section IV includes the related work that reviews operational profiles. Finally, Section V concludes this research.

II. Systematic Literature Review Process

To conduct this SLR, three guidelines are followed [1-3]. These guidelines provide the steps to conduct the literature review. The SLR method of conducting a literature review is borrowed from clinical research to organize the data from previous research and systematically deducing the results. The sub-sections include these step by step details to conduct the research process. These steps are review protocol, framing research questions, the primary studies selection, search keyword selection, inclusion and exclusion criteria for primary studies and results and synthesis based on selected primary studies. Initially few papers were handpicked seeing the titles and abstracts. Then, a citation based on forwarding snowballing strategy was adopted [16], computing inclusion and exclusion criteria and examining search statistics of the focused domain. In the subsequent stage, specialized search queries were formed to gather the studies that satisfied the inclusion-exclusion criteria and their match relevance.

A. Review Protocol

The SLR review protocol helps us to execute this research process with necessary actions and outputs. The SLR research protocol is shown in Fig. 2. The SLR process is started to provide the rationale for the purpose and need of study. The research questions are framed to collect the data for the purpose of fulfilling the research objectives. The next step is to collect the primary studies, inclusion and exclusion criteria, which helps to collect the most relevant research studies with respect to the research questions framed for this SLR. The data extraction method is devised to collect data from primary studies and then finally the data has been collected for synthesis and analysis purpose.

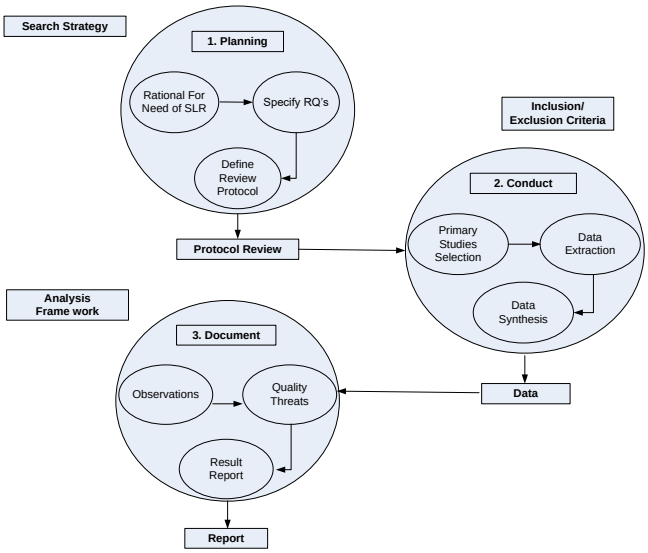

Fig. 2. The Review Protocol for Systematic Literature Review.

B. Research Questions

The research questions are framed with the help of discussions with the domain experts and software testing literature blind searches. The primary focus of these research questions is to find out the most relevant research on regression based test case prioritization adequacy criterions for test case prioritization, datasets available and used in controlled experiments for test case prioritization, measurements, and metrics available for regression testing and test case prioritization. The focus of these research questions was also to find those test case prioritization techniques which use more than one or two prioritization parameters and the effects of these parameters on the results of these techniques. The SLR also tries to focus on effectiveness as a measurable fact which so far discussed in the literature as a qualitative fact instead of a quantitative parameter [3, 12, 13]. The research questions are shown in Table I, with their justification to include in this SLR.

C. The Study Selection Procedure

The most important part of an SLR is its selection of primary studies which provides the ground for synthesis and analysis of data. The objective is to collect the most relevant data for results that identify the domain trends and dominant research problems with their solution space [3]. The quality of results based on the relevance of these primary studies. The selection of primary studies for this SLR based on the following steps.

- The selection of research repositories.
- The formulation and choice of keywords for search queries.
- The inclusion and exclusion criteria for searched studies with respect to the research questions.

TABLE I. THE RESEARCH QUESTIONS FOR SYSTEMATIC LITERATURE REVIEW

No	Research Questions	Justification
RQ-1	What is the state of the art research in regression TCP types/techniques?	The objective of this research question is to identify the important trends in the regression TCP research domain in order to collect the evidence for design and analysis for new and emerging regression TCP techniques.
RQ-2	What are the selection parameters used in regression TCP techniques?	The objective of this research question is to identify all possible selection parameters for regression TCP techniques and to find why they are used as selection criteria for test case selection.
RQ-3	What type of datasets used in regression TCP experiments?	The purpose of this research question is to find out the datasets for regression TCP experimentation and their usage.
RQ-4	What type of metrics/ evaluation criterions are used to verify the regression TCP techniques?	This research question helps to identify the possible metrics to evaluate and verify the regression TCP techniques and methods.

1) The selection research repositories: The process to identify the primary studies has been initiated by randomly entering the search keywords to research repositories. These retrieved research studies are then compared to the objectives of the research questions and inclusion/ exclusion criterion has been applied to these retrieved research studies. The choice of research repositories is quite important because of the quality dependent on these choices. For this purpose, in mind, the authors used the following research repositories is used for this process.

a) Science Direct.

b) IEEE Explore.

c) ACM Library.

The choice of these repositories based on the fact that IEEE Explore and ACM Library contains almost every important conference in the software testing domain. The Science Direct contains the research studies of almost all important journals relevant to the software testing research domain [4, 5].

2) Search keywords selection: A precise and systematic approach has been devised to search the search keywords. The approach is comprising of the following steps.

a) The most repeated keywords are selected from review papers on software testing and regression testing.

b) Find out the matching words, alternative keywords, similar words for these most frequently used terms in software testing literature.

c) Then devised search strings and search queries by using AND, OR and NOT operators available in research repositories search engines.

d) In the last step, we apply manual verification on searched studies that the research studies are relevant to the research questions.

In order to collect the most relevant research studies, authors try to switch the keywords with OR operator with author titles and author keywords are switched. The time period is also defined from 2007 to 2019 to limit the number of studies and covering the last twelve years of progress in the domain. This time limit was applied to the reason that software testing has a tremendous amount of research papers, but the systematic methodology was adopted in the year 2007, so it is helpful to limit the most relevant studies by applying this time limit. The search queries are shown in Table II.

TABLE II. THE SEARCH QUERY FOR SYSTEMATIC LITERATURE REVIEW

Repository	Search Query
IEEE	((((((("Publication Title":test case prioritization) OR "Abstract":test case prioritization) OR "Author Keywords":test case prioritization) OR "Publication Title":test suite prioritization) OR "Author Keywords":test suite prioritization) OR "Abstract":test suite prioritization) Filters Applied: Conferences Journals 2007 - 2019
ACM	"query": { acmdlTitle:(+Test +case + prioritization) OR recordAbstract:(+Test +case + prioritization) OR keywords.author.keyword:(+Test +case + prioritization) "filter": {"publicationYear":{ "gte":2007 }}, {owners.owner=HOSTED}
Web of science	TITLE: (Test case prioritization) OR TITLE: (Test suite prioritization) Timespan: years 2007. Indexes: SCI-EXPANDED, SSCI, A&HCI, CPCI-S, CPCI-SSH, BKCI-S, BKCI-SSH, ESCI.

3) Inclusion and exclusion criteria for searched studies: The regression test case prioritization has many different objectives with application domains, testing scope and testing environments. The test case prioritization in general considered as test suite optimization technique, but it is also observed that optimization research has many viewpoints, applications other than software testing. The challenge in study selection was the diversity of the topics covered under software testing such as software test suite prioritization, reduction, and augmentation. The experimental scope and size is also the main concern while selecting primary studies. As it was stated that the focus of this SLR was to collect the evidence for research studies considering cost, coverage and fault detection ability as test case prioritization criteria and effectiveness of the proposed techniques must be considered as one of the objectives of these studies. Therefore, it was required to design some rules while including or excluding the searched studies. Here, the inclusion and exclusion criteria applied to search studies were discussed.

a) The search queries are applied to selected research repositories and found 855 research studies. Then the authors applied two-stage inclusion criteria as shown in Table II.

b) The studies must be in the English language.

c) On the first level, the studies selected which have test case prioritization, test suite optimization with test case prioritization, test suite effectiveness, cost/coverage/fault

detection based test case prioritization in their title are selected.

d) The studies not included test case prioritization, test suite effectiveness or test case prioritization with some optimization criteria in their title or abstract are excluded.

e) The research studies that are not experiments, controlled experiments, case studies or without empirical results are also excluded.

Table III presents a two-Stage Spectrum of Research Studies Inclusion/Exclusion.

TABLE III. THE TWO STAGE SPECTRUM OF RESEARCH STUDIES INCLUSION/EXCLUSION

Research DB	First Searched Studies	First Round Exclusion	Second Round Inclusion
Science Direct	135	23	15
ACM Library	623	108	8
IEEE Explorer	900	260	29
Total	1658	391	52

After the first level of exclusion/inclusion, the authors started the second level of exclusion/inclusion. In this phase the studies are organized as per the research question framed. The content of each research study is compared with the objectives of the research questions especially the experimental process and result section of the study. The studies are now excluded/included based on the following rules.

1) The studies that did not report any experimental, case study or controlled experimental results are excluded.

2) The studies less than five pages and without experimental details are excluded.

3) The posters, PhD or Master thesis are excluded.

4) The technical reports are excluded.

5) The studies that did not focus on test case prioritization, test case prioritization optimization is excluded.

6) The studies that did not consider cost, coverage and fault detection ability as prioritization criteria or effectiveness criteria are excluded.

The purpose of the second phase of exclusion was to collect the most relevant and reasonably high-quality research studies with some experimental insights towards the domain. After second phase of inclusion/exclusion, authors left with fifty-two research studies that focus on regression test case prioritization with focus on cost, coverage or fault detection ability as test case prioritization parameters or used as effectiveness measure from these three parameters (cost, coverage and fault detection ability).

D. The Data Collection Strategy

After collecting the most relevant studies from inclusion/exclusion criteria, the data collected from these studies have been followed [6, 7]. The data also collected into two phases. The first phase consists of a study title, publication year and source, summary of the research study and comments of the researcher. In the second phase, the technical information with respect to the research questions has been collected to answer the research questions framed for this SLR. The first phase data collection helps the researchers to execute the inclusion and exclusion phase. The second phase of data collection helps the researchers to synthesis and analysis the results of this SLR.

III. RESULTS AND DISCUSSION

In this section, the results are presented based on the data collected from the primary studies to answer the research questions framed in the previous section. There were four questions framed for synthesis and analysis. These questions were framed to identify the main research gaps and important trends in the development and design of the regression test case prioritization research domain. The second focus was to identify the datasets and experimental evaluation trends and features in the regression test case prioritization research domain. The results for Research Question 1: The state of the art research in each research questions are as followed regression test case prioritization types/techniques.

The objective of this research question was to assess the state-of-the-art research conducted in the domain of regression test case prioritization with the focus on cost, coverage and fault detection ability. The analysis performed on the data collected from primary studies shown the following regression test case prioritization techniques families as in Fig. 3. Each technique has common input, processing, and output styles but differs in their designing parameters and context of usage.

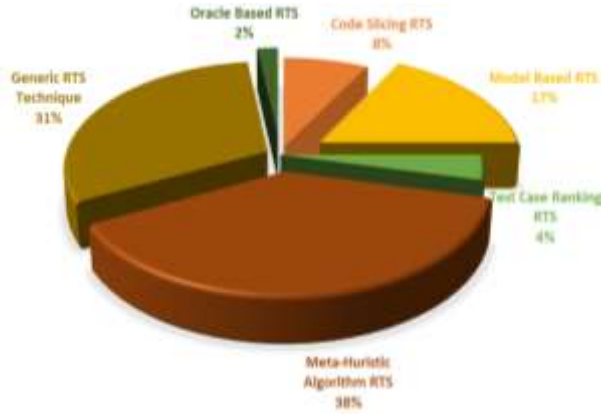

Fig. 3. The RTS Techniques Classification based on Primary Studies.

From Fig. 3, the major families of regression test case selection techniques as follows.

1) The meta-heuristic based TCP
2) Model-based TCP
3) Generic Based TCP techniques.
4) The test case ranking based TCP.
5) The Code slicing based TCP.
6) The Oracle Based TCP.

The meta-heuristic based TCP techniques were found 38%, as a leading trend in TCP methods. There were 20 out of 52 studies that used these algorithms to solve or implement the solution for the TCP problems. The reason for its popularity was its capability to handle the multi-criteria problems with emerging tools and technologies for analysis and design for

these algorithms. From these meta-heuristic family Genetic Algorithm (GA) was observed in seven (7) out of twenty (20) studies and become the most widely used algorithm for TCP problems. The GA is considered as evolutionary optimization technique with the inbuilt believe in survival to the fittest. The GA is popular for TCP solution design due to its nature of selection the stronger population based on some fitness function. The design and process of GA very much like TCP design and process of selection. TCP selects the test cases to prioritize from already used test suites based on some criteria while GA selects the stronger population from previous populations based on fitness functions. The second reason for the choice of GA for TCP problems was its maturity and there were so many comparative studies available for this algorithm. There are many datasets available with evaluation metrics with GA in the test case prioritization research domain. The different types of GA used in these studies are the Co-evolutionary Genetic Algorithm (CGA), Diversity Based Genetic Algorithm (Div-GA), Multi-Objective Genetic Algorithm (MOGA) and Non-Dominated Sorting Genetic Algorithm (NSGA-II).

The second most used algorithm in TCP problem solving was Particle Swarm Optimization (PSO) observed in five (5) studies out of twenty (20) studies. PSO is a greedy algorithm that tries to find a local maximum from the problem space. It is easy to implement as compared to GA. But the choice between GA and PSO depends on the nature and design of the problem. The different types of implementations of PSO from primary studies are simple PSO, Multi-objective PSO and Additional Greedy based on voting mechanism PSO.

The fuzzy algorithm is the third most used algorithm four (4) out of twenty studies. The fuzzy is used with types of rule-based fuzzy, fuzzy classification and fuzzy expert system. The fuzzy is quite a simple but static decision-making system. The prior defined rules are used to decide the different decisions required during the selection of TCP. The K-means and semi-supervised clustering also used in two different studies for TCP problems.

The second class of solutions for TCP problems were Generic TCP solutions found in 31% of the studies. There were sixteen (16) out of fifty-two studies (52) studies that used these methods, tools, and algorithms. These are self-designed custom solutions for specific tools and problems. Normally they are applied to industrial-scale case studies to solve TCP problems.

Model-based TCP is the third popular class of TCP solutions. It was used in nine (9) studies out of fifty-two (52) studies. It is based on Unified Modeling Language (UML) artifacts to prioritize the test cases for software under testing. The used artifacts were activity diagrams, state machines, and use case diagrams. But these diagrams appear so early in the software life cycle, so, they are so much imprecise to use as test case prioritization solutions. The code slicing and chopping techniques are used in four (4) studies out of fifty-two (52) studies, 8% of the total studies. These techniques were relevant due to code modifications are the primary focus of regression TCP techniques. The code changes and modifications are easy to identify by code slicing and code chopping techniques. But due to the complexity of new coding environments, it is difficult to chop the code with modern code editors and code generators. The test case ranking regression TCP techniques were seen in two out of fifty-two studies. The test cases and their results were used to rank the test cases for future use in these techniques.

A. *Research Question 2 The Selection Parameters used in RTP Techniques*

This research was framed to identify the number of parameters used for test case prioritization techniques. The objective was to understand the fact that available space for research in designing new test case prioritization techniques, their design trends and the dependency among these parameters if there is any dependency among these parameters. The well-known parameters are cost, coverage and fault detection ability [8]. The definitions of these measures are as following.

1) Cost: The time or resources consumed by a test suite/ test case to complete its execution on source code to return its results. The further types of cost observed are time to run a test suite, time to create a test suite, time to analysis for a test suite and time to prepare the results of a test suite.

2) Code coverage: The ratio of source code executed by a test case/test suite to the total number of source lines expected to execute by that test suite/test case is known as code coverage. Its special sub-groups are statement coverage, condition coverage, modified condition coverage, loop coverage, branch coverage, modified branch coverage, and modified statement coverage.

3) Fault detection ability: The number of the faults identified by a test suite/test case is known as fault detection ability of that test suite/test case. The sub-types of faults observed in primary studies are structural faults, real faults, hand seeded faults and mutation faults.

The results of the research question are shown in Fig. 4 below. The observed classes of these prioritization criteria are cost, code coverage and fault detection ability as single criteria to test case prioritization techniques. The code coverage with Fault detection ability and cost and code coverage are observed as bi-criteria test case prioritization parameters. The cost, code coverage, and fault detection ability are observed as tri-criteria test case prioritization parameters.

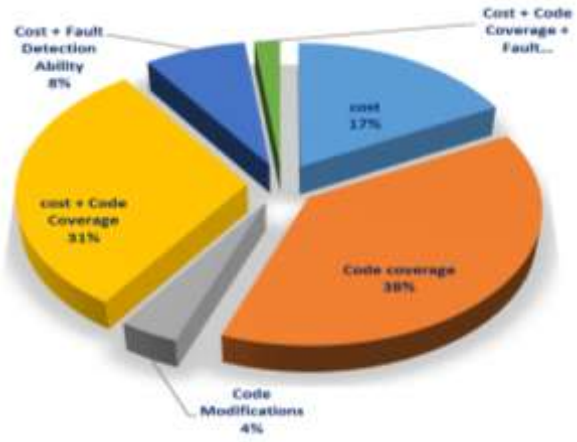

Fig. 4. Test Case Prioritization Parameters Classification based on Primary Studies.

The code coverage is the most dominant trend observed in selected primary studies. It was found in twenty (20) out of fifty-two (52) primary studies which were 38% of the total primary studies. The reasons for using code coverage were its simple computation with respect to other parameters. There were a good number of the tools available for measuring code coverage of different types and its integration is quite simple with available code editors and code generators like Eclipse, Junit, Code Cover, Mue-java, etc. The measurement of code coverage is simple enough and decision making is also very straight forward. The more code coverage provides more confidence in testing teams that their code is tested. The code coverage is used as a proxy in many test scenarios which means for testing teams, quality assurance groups, management teams and customers of the product.

The second dominant test case prioritization parameter group was cost and code coverage, which is also a bi-criteria test case prioritization family. It was observed in sixteen (16) out of fifty-two (52) primary studies, which was 31% of the total primary studies. The reason for this was in close resemblance in the measurement of these parameters. Both code coverage and cost metrics returned the results in measurable numbers. The available tool support for measuring cost and code coverage. The code coverage and cost measurement both dependent on each other, more coverage means more cost for a test suite. The more cost means a less effective regression TCP technique. Both code coverage and cost were primary objectives for the optimization of test case prioritization techniques.

The third trend found in primary studies was cost-based test case prioritization techniques. It was found in nine (9) out of fifty-two (52) primary studies which were 17% of the total primary studies. The optimization of the cost was the primary objective of a test case prioritization technique because the reduction in cost means a better test case prioritization technique which may replace the previous regression TCP technique. The cost measures are observed with many different viewpoints like execution cost of a test suite, size of the source code under testing, size of the test cases in a test suite, analysis time for results of a test suite, preparation of the test suite for a software under testing and post-analysis and prioritization time of a test suite. The choice of cost measures depends on the local requirements of optimization of test case prioritization problems. The fourth trend was the tri-criteria test case prioritization parameter comprises of three measures cost, code coverage, and fault detection ability. The combination of these three parameters makes test case prioritization more effective because fault detection ability is the primary objective of all software testing techniques. The fault detection ability in test case prioritization techniques used as adequacy criteria so far, but in a few techniques, it was used as test case prioritization parameters as well. The tri-criteria optimization seen as a challenge in test case prioritization problems due to the huge size of the code and test suite sizes for software under testing. The last test case prioritization parameter was code modifications, the identification of code changes from code in code chopping techniques. The code chopping was not practical due to increase in size and complexity of the source code in modern software. The second reason was the security and safety requirements of the third-party source codes which may not provide direct access to the critical pieces of the source codes.

B. *Research Question 3: Datasets used in Regression TCP Experiments*

The research question was framed to identify the datasets used in software testing experimentation with a special focus on designing the novel techniques for regression testing. The software testing datasets are quite different in nature as compared to other artifacts used in software engineering research. The point of differences and important features considered during datasets for regression testing are as follows continue Table IV.

1) There must be a reasonable source code size for the software under testing.

2) There is must be a test suite available for testing with previous testing cycle's history or results which justify the usage of that test suite.

3) There must be some tool/framework/methodology support available to execute that testing technique on software under testing.

4) There should be some measurement mechanism to evaluate and compare the results for that testing technique.

5) The source code and test suite collections must available to other research communities to use as an artefact for their experiments.

6) The results and conclusions must be based on some environment available to other research communities to evaluate and compare with their findings with the previous research findings.

There were three sources identified providing software source code, test suites, test results and tool information used to collect the results for software testing experiments. These sources are as follows:

1) SIR (Software Artefact Infrastructure Repository).
2) Open Source (Apache Software Foundation).
3) Git-Hub.

The Software-Artefact Infrastructure Repository (SIR) [9] is the collection of software source codes with multiple versions and associated test suites. It has the artifacts that have a wide range of software with many different programming languages like Java, C, C++, PHP and C-sharp. These datasets are prepared for unit testing, integration testing, system testing. The fault types supported by these datasets are real faults, hand seeded faults and mutation faults [9]. The detailed primary studies and subject software are listed in Table IV.

The second repository which offers a wide range of datasets for software testing artifacts is Apache Software Foundation [61]. This repository contains 200 Million lines of source code and 350 projects with multiple versions of source code and test suites for each version. The Git Hub is also a very huge size code repository for software source codes and their test suites. The third repository Git Hub [62] is a general-purpose repository in which individual developers and software engineers upload their code and test suites. The choice of datasets depends upon the nature and design of the problem,

available tool support for technique under analysis and measurement methods used to evaluate results produced with these datasets.

TABLE IV. THE DATASETS IDENTIFIED FOR RTS EXPERIMENTS IN PRIMARY STUDIES

Study	Reference	Dataset
1	[10]	Custom Product/Code
2	[11]	Custom Product/Code
3	[12]	SIR (Siena, Jtopas)
4	[13]	Not Reported
5	[14]	Not Reported
6	[15]	Custom Product/Code
7	[16]	Aspect Compiler example package = 3 programs
8	[17]	ABB = 3 programs, SIR = 2 programs
9	[18]	ABB program
10	[19]	Custom Product/Code
11	[20]	SIR (Jmeter), XML(Security, ANT)
12	[21]	Custom Product/Code
13	[22]	Custom Product/Code
14	[23]	SIR (Flex, Space, Schedule)
15	[24]	Student Enrolment System.
16	[25]	SIR (Nano,ant,Galileo, Jmeter,XML)
17	[26]	SIR (nanoXML,jtops,jmeter,xml- security, any)
18	[27]	Custom Product/Code
19	[28]	Safety Monitoring Component
20	[29]	Open-Source (Apache,Log 4j, common-Math)
21	[30]	Open-Source (Polo)
22	[31]	SIR(Space)
23	[32]	Custom Product/Code
24	[33]	Video-conference system Safety Monitoring Components
25	[34]	Not Reported
26	[35]	Microsoft Dynamics AX
27	[36]	Not Reported
28	[37]	Custom Product/Code
29	[38]	Scheduler
30	[39]	SIR(print tokens, printtokens1, scheduler, scheduler2, space)
31	[40]	Custom Product/Code
32	[41]	SIR 11 programs
33	[42]	SIR(printtokens, printtokens2)
34	[43]	Custom Product/Code
35	[44]	SOFIE is the tax accounting system
36	[45]	Custom Product/Code
37	[46]	Custom Product/Code
38	[47]	Not Reported
39	[48]	SIR (space)
40	[49]	Calendar, triangle, time-date,Kmap generation, tax calculation.
41	[50]	Custom Product/Code
42	[51]	Custom Product/Code
43	[52]	Custom Product/Code
44	[53]	11 Large Open-source projects
45	[54]	61 open source systems
46	[55]	11 open-source projects
47	[56]	21 Java projects
48	[57]	Grep v1 to v7
49	[58]	Custom datasets for 3 sprints
50	[59]	JFree Chart, Apache Tomcat, Argo UML
51	[60]	37 projects on Git Hub

C. *Research Question 4: Type of the Metrics/ Evaluation Criterions are used to Verify the Regression TCP Techniques*

The research question was framed to identify the measurements and methods to evaluate the results collected from regression TCP experiments. The important features are those which classify the effectiveness abilities of one technique to another technique. There are so many different viewpoints observed from primary studies collected for this SLR. The notable trends were comparing the results in terms of their input, process and output styles, their method to prioritize the test cases, their ability to identify the faults and fault types and their presentation method of the finding for analysis performed on datasets.

The code-based regression TCP techniques primarily designed to reduce the cost of testing in terms of execution time, test suite size and try to satisfy the code coverage required criteria and cost evaluation and fault detection ability. In the evaluation of regression TCP experimental results, a framework has been proposed in the study [63]. This evaluation structure classifies the test suites in the following types.

1) Obsolete Test Cases: A test case that uncovers nothing new like faults or code modifications.

2) Modification Revealing: A test case that executes a modified part of the code under testing.

3) Non-Modification Revealing Test Case: A test case that does not execute a modified part of the code under testing.

4) Fault Revealing Test Cases: The test cases which identify the faults from the source code under testing.

The other metrics identified from the primary studies are inclusiveness, precision, fault measure, fault rate, code coverage, fault metrics, and retest-all. The studies are compared in terms of effectiveness, cost, code coverage, and fault detection ability. These metrics are mentioned because there were proper mathematical grounds for these metrics were available. The second reason was that many experiments reported from primary studies may be useful to compare the results with future studies.

IV. Conclusion and Future Work

The research study was conducted by following a systematic literature review methodology. The review protocol was designed and conduct the search from relevant research repositories with the research questions framed in the review protocol. There were 1658 studies found from three research repositories. There were two-stage inclusion/exclusion criteria to choose the most relevant studies with respect to the research questions. There were 391 studies left on the first level of inclusion/exclusion criteria. On the second level of inclusion/exclusion criteria, there were fifty-two studies left. The analysis of primary studies reveals that there are six main classes of test case prioritization techniques such as meta-heuristic regression TCP, code slicing regression TCP, model-based regression TCP, test case ranking regression TCP, Oracle-based regression TCP, and Generic regression TCP techniques. The regression TCP parameters have cost, coverage and fault detection, as single criteria regression TCP, the cost and coverage and fault and coverage as bi-criteria regression TCP and cost, coverage and fault detection as tri-criteria regression TCP techniques. There was a long list of datasets available for controlled experiments of regression testing experiments. The main sources to obtain these datasets were SIR, Git Hub, and Open Source Apache Software Foundation. It is also concluded that meta-heuristic techniques are the most researched trend so far. The genetic algorithm was the most used algorithm for regression TCP solutions. The code coverage is the most used parameter for test case prioritization. Based on these results, the authors recommend that more experimental research is required to investigate bi-criteria and tri-criteria test case prioritization techniques. It is also concluded that cost, coverage, fault detection ability and code modifications are equally important for selecting a test suite for software under testing. The results reviewed from primary studies show that these studies ignore one or two prioritization parameters. It is also observed that the local constraints like tools, programming languages, and measurement and metrics also need to be researched experiment to produce generalized results for the whole testing community.

References

[1] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," Journal of systems and software, vol. 80, pp. 571-583, 2007.

[2] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in Proceedings of the 28th international conference on Software engineering, 2006, pp. 1051-1052.

[3] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," Information and software technology, vol. 51, pp. 7-15, 2009.

[4] M. Younas, D. N. Jawawi, I. Ghani, T. Fries, and R. Kazmi, "Agile development in the cloud computing environment: A systematic review," Information and Software Technology, vol. 103, pp. 142-158, 2018.M. Khatibsyarbini, M. A. Isa, D. N. 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.

[5] B. Kitchenham, H. Al-Khilidar, M. A. Babar, M. Berry, K. Cox, J. Keung, et al., "Evaluating guidelines for reporting empirical software engineering studies," Empirical Software Engineering, vol. 13, pp. 97-121, 2008.

[6] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, et al., "Preliminary guidelines for empirical research in software engineering," IEEE Transactions on Software Engineering, vol. 28, pp. 721-734, 2002.

[7] M. Huang, S. Guo, X. Liang, and X. Jiao, "Research on regression test case selection based on improved genetic algorithm," in Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on, 2013, pp. 256-259.

[8] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," Empirical Software Engineering, vol. 10, pp. 405-435, 2005.

[9] C. Tao, B. Li, X. Sun, and C. Zhang, "An approach to regression test selection based on hierarchical slicing technique," in Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual, 2010, pp. 347-352.

[10] W.-T. Tsai, X. Zhou, R. A. Paul, Y. Chen, and X. Bai, "A coverage relationship model for test case selection and ranking for multi-version software," in High Assurance Systems Engineering Symposium, 2007. HASE'07. 10th IEEE, 2007, pp. 105-112.

[11] C. Tao, B. Li, X. Sun, and Y. Zhou, "A hierarchical model for regression test selection and cost analysis of java programs," in Software Engineering Conference (APSEC), 2010 17th Asia Pacific, 2010, pp. 290-299.

[12] W. S. A. El-hamid, S. S. El-etriby, and M. M. Hadhoud, "Regression test selection technique for multi-programming language," in 2010 The 7th International Conference on Informatics and Systems (INFOS), 2010, pp. 1-5.

[13] S. Huang, Z. J. Li, J. Zhu, Y. Xiao, and W. Wang, "A novel approach to regression test selection for J2EE applications," in 2011 27th IEEE international conference on software maintenance (ICSM), 2011, pp. 13-22.

[14] Z. Xu, Y. Liu, and K. Gao, "A novel fuzzy classification to enhance software regression testing," in Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on, 2013, pp. 53-58.

[15] G. Xu and A. Rountev, "Regression test selection for AspectJ software," in Software Engineering, 2007. ICSE 2007. 29th International Conference on, 2007, pp. 65-74.

[16] T. Yu, X. Qu, M. Acharya, and G. Rothermel, "Oracle-based regression test selection," in Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on, 2013, pp. 292-301.

[17] J. Zheng, L. Williams, B. Robinson, and K. Smiley, "Regression test selection for black-box dynamic link library components," in Proceedings of the Second International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques, 2007, p. 9.

[18] P. K. Chittimalli and M. J. Harrold, "Recomputing coverage information to assist regression testing," Software Engineering, IEEE Transactions on, vol. 35, pp. 452-469, 2009.

[19] N. Rachatasumrit and M. Kim, "An empirical investigation into the impact of refactoring on regression testing," in Software Maintenance (ICSM), 2012 28th IEEE International Conference on, 2012, pp. 357-366.

[20] A. Pasala, Y. L. Y. Fung, F. Akladios, G. A. Raju, and R. P. Gorthi, "Selection of regression test suite to validate software applications upon deployment of upgrades," in 19th Australian Conference on Software Engineering (aswec 2008), 2008, pp. 130-138.

[21] E. Fourneret, J. Cantenot, F. Bouquet, B. Legeard, and J. Botella, "Setgam: Generalized technique for regression testing based on uml/ocl models," in 2014 Eighth International Conference on Software Security and Reliability (SERE), 2014, pp. 147-156.

[22] A. Nanda, S. Mani, S. Sinha, M. J. Harrold, and A. Orso, "Regression testing in the presence of non-code changes," in 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation, 2011, pp. 21-30.

[23] S. Chen, Z. Chen, Z. Zhao, B. Xu, and Y. Feng, "Using semi-supervised clustering to improve regression test selection techniques," in Software Testing, Verification and Validation, 2011 IEEE Fourth International Conference on, 2011, pp. 1-10.

[24] M. Z. Z. Iqbal, Z. I. Malik, and M. Riebisch, "A model-based regression testing approach for evolving software systems with flexible tool support," in 2010 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems, 2010, pp. 41-49.

[25] S. Mirarab, S. Akhlaghi, and L. Tahvildari, "Size-constrained regression test case selection using multicriteria optimization," IEEE Transactions on Software Engineering, vol. 38, pp. 936-956, 2012.

[26] Y. Pang, X. Xue, and A. S. Namin, "Identifying effective test cases through k-means clustering for enhancing regression testing," in Machine Learning and Applications (ICMLA), 2013 12th International Conference on, 2013, pp. 78-83.

[27] L. S. de Souza, R. B. Prudêncio, and F. d. A. Barros, "A Hybrid Binary Multi-objective Particle Swarm Optimization with Local Search for Test Case Selection," in Intelligent Systems (BRACIS), 2014 Brazilian Conference on, 2014, pp. 414-419.

[28] H. Hemmati and L. Briand, "An industrial investigation of similarity measures for model-based test case selection," in 2010 IEEE 21st International Symposium on Software Reliability Engineering, 2010, pp. 141-150.

[29] H. Cibulski and A. Yehudai, "Regression test selection techniques for test-driven development," in Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, 2011, pp. 115-124.

[30] A. A. L. de Oliveira, C. G. Camilo-Junior, and A. M. Vincenzi, "A coevolutionary algorithm to automatic test case selection and mutant in mutation testing," in Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013, pp. 829-836.

[31] L. S. de Souza, P. B. de Miranda, R. B. Prudencio, and F. d. A. Barros, "A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort," in 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, 2011, pp. 245-252.

[32] H. Hemmati, A. Arcuri, and L. Briand, "Empirical investigation of the effects of test suite properties on similarity-based test case selection," in 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation, 2011, pp. 327-336.

[33] M. E. Delamaro and J. Offutt, "Assessing the influence of multiple test case selection on mutation experiments," in 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops, 2014, pp. 171-175.

[34] J. Anderson, S. Salem, and H. Do, "Improving the effectiveness of test suite through mining historical data," in Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 142-151.

[35] H. Hemmati, L. Briand, A. Arcuri, and S. Ali, "An enhanced test case selection approach for model-based testing: an industrial case study," Proceedings of the 8th ACM SIGSOFT international symposium on Foundations of software engineering, 2010, pp. 267-276.

[36] S. Huang, Y. Chen, J. Zhu, Z. J. Li, and H. F. Tan, "An optimized change-driven regression testing selection strategy for binary Java applications," in Proceedings of the 2009 ACM symposium on Applied Computing, 2009, pp. 558-565.

[37] H. Hemmati, A. Arcuri, and L. Briand, "Reducing the cost of model-based testing through test case diversity," in IFIP International Conference on Testing Software and Systems, 2010, pp. 63-78.

[38] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in Proceedings of the 2007 international symposium on Software testing and analysis, 2007, pp. 140-150.

[39] L. Yu, L. Xu, and W.-T. Tsai, "Time-constrained test selection for regression testing," in International Conference on Advanced Data Mining and Applications, 2010, pp. 221-232.

[40] A. Panichella, R. Oliveto, M. Di Penta, and A. De Lucia, "Improving multi-objective test case selection by injecting diversity in genetic algorithms," IEEE Transactions on Software Engineering, vol. 41, pp. 358-383, 2014.

[41] M. Kumar, A. Sharma, and R. Kumar, "Fuzzy entropy-based framework for multi-faceted test case classification and selection: an empirical study," IET software, vol. 8, pp. 103-112, 2013.

[42] Z. Xu, K. Gao, T. M. Khoshgoftaar, and N. Seliya, "System regression test planning with a fuzzy expert system," Information Sciences, vol. 259, pp. 532-543, 2014.

[43] E. Rogstad, L. Briand, and R. Torkar, "Test case selection for black-box regression testing of database applications," Information and Software Technology, vol. 55, pp. 1781-1795, 2013.

[44] L. S. De Souza, R. B. Prudêncio, F. d. A. Barros, and E. H. d. S. Aranha, "Search based constrained test case selection using execution effort," Expert systems with applications, vol. 40, pp. 4887-4896, 2013.

[45] B. Li, D. Qiu, H. Leung, and D. Wang, "Automatic test case selection for regression testing of composite service based on extensible BPEL flow graph," Journal of Systems and Software, vol. 85, pp. 1300-1324, 2012.

[46] Y.-D. Lin, C.-H. Chou, Y.-C. Lai, T.-Y. Huang, S. Chung, J.-T. Hung, et al., "Test coverage optimization for large code problems," Journal of Systems and Software, vol. 85, pp. 16-27, 2012.

[47] Z. Chen, Y. Duan, Z. Zhao, B. Xu, and J. Qian, "Using program slicing to improve the efficiency and effectiveness of cluster test selection," International Journal of Software Engineering and Knowledge Engineering, vol. 21, pp. 759-777, 2011.

[48] Y. Singh, A. Kaur, and B. Suri, "A hybrid approach for regression testing in interprocedural program," Journal of Information Processing Systems, vol. 6, pp. 21-32, 2010.

[49] N. Mansour, H. Takkoush, and A. Nehme, "UML-based regression testing for OO software," Journal of Software Maintenance and Evolution: Research and Practice, vol. 23, pp. 51-68, 2011.

[50] E. G. Cartaxo, P. D. Machado, and F. G. O. Neto, "On the use of a similarity function for test case selection in the context of model-based testing," Software Testing, Verification and Reliability, vol. 21, pp. 75-100, 2011.

[51] L. Zhang, "Hybrid regression test selection," in 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), 2018, pp. 199-209.B. Fu, S. Misailovic, and M. Gligoric, "Resurgence of Regression Test Selection for C++," in 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), 2019, pp. 323-334.

[52] A. Labuschagne, L. Inozemtseva, and R. Holmes, "Measuring the cost of regression testing in practice: a study of Java projects using continuous integration," in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 821-830.

[53] M. Vasic, Z. Parvez, A. Milicevic, and M. Gligoric, "File-level vs. module-level regression test selection for. net," in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 848-853.

[54] A. Celik, M. Vasic, A. Milicevic, and M. Gligoric, "Regression test selection across JVM boundaries," in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 809-820.

[55] B. Miranda and A. Bertolino, "Scope-aided test prioritization, selection and minimization for software reuse," Journal of Systems and Software, vol. 131, pp. 528-549, 2017.

[56] P. Kandil, 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, p. e1794, 2017.

[57] B. Guo and M. Song, "Interactively decomposing composite changes to support code review and regression testing," in 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC), 2017, pp. 118-127.

[58] K. Wang, C. Zhu, A. Celik, J. Kim, D. Batory, and M. Gligoric, "Towards refactoring-aware regression test selection," in 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), 2018, pp. 233-244.

[59] Apache. (2017). JodaTime. Available: http://www.joda.org/joda-time/

[60] GitHub. Software code Hosting [Online]. Available: https://github.com/

[61] G. Rothermel and M. J. Harrold, "A framework for evaluating regression test selection techniques," in Software Engineering, 1994. Proceedings. ICSE-16., 16th International Conference on, 1994, pp. 201-210.