SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Comprehensive Review on the State-of- the-arts and Solutions to the Test Redundancy Reduction Problem with Taxonomy

Mizanur Rahman[1], Kamal Z. Zamli[1,2], Md. Abdul Kader[1], Roslina Mohd Sidek[1,*], Fakhrud Din[3]

[1] Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia
[2] Faculty of Science and Technology, Universitas Airlangga, C Campus JI. Dr. H. Soekamo, Mulyorejo, Surabaya 60115, Indonesia
[3] Faculty of Information Technology, Department of Computer Science & IT, University of Malakand, Lower Dir 18800, KPK, Pakistan

ARTICLE INFO

Article history:
Received 9 June 2023
Received in revised form 3 October 2023
Accepted 13 November 2023
Available online 13 December 2023

Keywords:

Test redundancy reduction; Test case reduction; Software testing; Test suite reduction

ABSTRACT

The process of software testing is of utmost importance and requires a major allocation of resources. It has a substantial influence on the quality and dependability of software products. Nevertheless, as the quantity of test cases escalates, the feasibility of executing all of them diminishes, and the accompanying expenses related to preparation, execution time, and upkeep grow excessively exorbitant. The objective of Test Redundancy Reduction (TRR) is to mitigate this issue by determining a minimal subset of the test suite that satisfies all the requirements of the primary test suite while lowering the number of test cases. In order to attain this objective, multiple methodologies have been suggested, encompassing heuristics, meta-heuristics, exact algorithms, hybrid approaches, and machine-learning techniques. This work provides a thorough examination of prior research on TRR, addressing deficiencies and making a valuable contribution to the current scholarly understanding. The literature study encompasses a comprehensive examination of the complete chronology of TRR, incorporating all pertinent scholarly articles and practitioner-authored research papers published in English. This study aims to provide managers with valuable insights into the strengths and shortcomings of different TRR methodologies, enabling them to make well-informed decisions regarding the most appropriate approach for their specific needs. The primary objective of this study is to offer a comprehensive analysis of Test Result Reduction (TRR) and its consequential impact on mitigating expenses related to software testing. This study makes a valuable contribution to extant literature by elucidating the present state-of-the-art and delineating potential avenues for future research.

1. Introduction

In software development, the testing phase is a crucial component to ensure the quality and reliability of software products. However, it is also one of the most time-consuming and expensive processes in the software development life cycle [1]. Testing involves the use of test cases to detect and identify defects in software. Test cases are created with specific inputs, execution steps, and

* *Corresponding author.*
E-mail address: roslinams@ump.edu.my

https://doi.org/10.37934/araset.35.1.6287

predicted results to test various software functions and ensure compliance with required specifications. According to Yang, and Li in [2], a bulk of test inputs, actions, and anticipated outcomes are created with a specific goal in mind, like exercising a specific software route or ensuring compliance with required specifications called a test case. A test suite is a collection of test cases used to verify the integrity of the program's source code and confirm that the system behaves correctly in all possible ways.

As software products evolve, the size of the test suite grows, and the cost of testing increases exponentially. Performing all possible test cases becomes impractical, and this is where Test Redundancy Reduction (TRR) methods come into play. TRR aims to capture a representative subset of the entire test set to ensure that all test requirements are met while avoiding unnecessary repetition. Reducing test redundancy can significantly reduce the time and cost of testing while still ensuring software quality and reliability [3]. TRR approaches aim to capture a representative subset of the entire test set to ensure that all test requirements are met while avoiding unnecessary repetition. It is claimed that a test case is redundant if there are additional test cases that can satisfy the same requirements. One approach to TRR is the set coverage problem, which is defined as an NP-complete problem. Johnson [4] was the first to propose a set covering approach, and since then, researchers have proposed various methods, including heuristic, exact, meta-heuristic, and machine learning approaches. Test redundancy reduction (TRR) is a critical problem in software testing as it directly impacts the efficiency and effectiveness of the testing process.

To address this gap, this paper aims to investigate and evaluate the current state-of-the-art TRR approaches, and performance metrics used in the test reduction process. The review will include research articles related to the TRR problem, and we will categorize the relevant work to provide a clear overview of the existing literature. This study will also identify the strengths and weaknesses of current TRR approaches and performance metrics and provide insights into future TRR algorithms and methods. Unlike previous review articles [5-7] that mainly focused on conference papers and were published before 2018, this review will include recently published journal articles, providing a more comprehensive and up-to-date assessment of the field. This research makes several significant contributions to the field of test redundancy reduction, including:

i. We provide a thorough evaluation of the most popular approaches for reducing test suites, including exact, heuristic, meta-heuristic, hybrid, and machine learning-based methods. Our analysis covers the entire history of test redundancy reduction and helps readers make informed decisions about which technique is best suited for their specific needs.
ii. We conduct a comprehensive analysis of the test redundancy reduction problem, segmenting and classifying existing solutions according to a newly developed taxonomy. Our taxonomy enables a clearer understanding of the various approaches to test suite reduction, helping researchers and practitioners identify gaps in the current literature and develop more effective methods.
iii. We offer a detailed assessment of the advantages and limitations of each approach to test suite reduction, providing readers with a comprehensive overview of the state of the art in this field. Our analysis also highlights areas for future research and development, helping to guide the direction of future work in this area.

The paper is organized as follows. Section 2 provides an overview of the test redundancy reduction problem, including its definitions, significance, and current state of research. Section 3 presents a novel taxonomy of test redundancy reduction approaches. This taxonomy categorizes and

organizes existing literature based on their underlying methodology, assumptions, and limitations. Section 4 includes a brief discussion of the study's findings and limitations. The conclusion, in section 5, summarizes the study's main contributions, implications, and suggestions for future research in the field of test redundancy reduction.

2. Overview of the TRR Problem

Software testing is a critical procedure that involves the assessment of a program's functioning in relation to predetermined requirements. In general, test engineers are responsible for creating a set of test cases in order to assess software in accordance with user demands and ascertain its compliance with predetermined standards. The escalating intricacy of software and its extensive use across varied industries necessitate the expertise of proficient engineers possessing a broad range of knowledge to effectively oversee software development endeavours. In the contemporary day, it is not atypical for software development teams to engage in collaborative efforts across diverse locales or regions. The complex nature of interdependent processes presents challenges in coordinating tasks, mostly due to the limitations imposed by spatial and temporal constraints on these interactions. A software test suite has the potential to have numerous test cases that are similar in nature, resulting in redundancies within the testing modules. As an illustration, it is possible to conduct many tests to fulfil a single need, hence resulting in an escalation of the overall testing expenditure. The authors employ many names to refer to the process of reducing redundancy in testing, including Test Suite Minimization and Test Suite Reduction. Throughout the remainder of this article, the term Test Redundancy Reduction will be employed. The objective of TRR is to generate a compact subset of the test suite that adequately fulfils the specified test requirements. The objective of this procedure is to decrease the size of the test suite while ensuring that it satisfies the test criteria, hence diminishing the overall cost of testing.

Table 1 shows that there are numerous occasions where one requirement is addressed by multiple test cases, leading to redundancies. For instance, t1 and t6 cover Req1. All t2, t3, t4, and t5 satisfy Req2. Similarly, t1, t3, and t4 cover Req3. A similar analysis may be performed for the remaining requirements, Req4 through Req9. In this instance, it is preferable to have fewer test cases while covering as many requirements as possible. The minimum set of tests for this issue is t1, t2, t4 or t1, t2, t5, both of which result in a 50% reduction, as may be inferred from Table 1. It should be mentioned that while manual inspection can be simple for small software applications with few tests and requirements, it is not scalable for large-scale software.

Table 1
Test Suite Scenario

	Req1	Req2	Req3	Req4	Req5	Req6	Req7	Req8	Req9
t1	X		X			X			X
t2		X		X	X			X	
t3		X	X			X			X
t4		X	X		X		X		
t5		X		X	X		X		
t6	X					X			

Mathematically, the test redundancy reduction problem can be represented as follows:

i. Define a set of all tests: Let T = {T_1, T2, ..., Tn} represent the set of all tests, where Ti is an individual test.

ii. Define a set of requirements: Let R = {(req_1, req_2,…, req_m)} represent the set of requirements that must be covered to produce the desired test coverage.

The objective function is to minimize the number of tests in T' (i.e., minimize ∑Ti).

$$f(t) = min\{Tm \text{ covering Req }\}, t = t1, t2 \dots tj; j = 1,2, \dots N \quad (1)$$

2.1. Research Questions

The research question is a critical component of any research project, as it provides a clear focus and direction for the study. It helps researchers to define the purpose and scope of the investigation, as well as to identify the appropriate methodology and data analysis techniques. A well-formulated research question should be clear, concise, and specific, and should be designed to address a particular gap in knowledge or understanding in a particular field of study. In this research paper, through the following three research questions, the researcher seeks to uncover new insights and contribute to the body of knowledge in the field.

RQ#1: What methods are available for the reduction of the test suite?
RQ#2: How do the current methods reduce the test suites?
RQ#3: How do scientists evaluate their test reduction experiments?

2.2 Research Method and Search String

To determine the total number of papers on test redundancy reduction (TRR), we downloaded all relevant papers and analysed the number of TRR papers published each year, categorizing them by the journals in which they appeared. Figure 1 shows the frequency with which papers on TRR are published in different journals. Most papers, 41%, were published by IEEE, followed by Springer with 20%, Elsevier with 18%, ACM with 13%, and others with 8%. Figure 1 shows that IEEE has the highest number of published articles. Our searches began by retrieving all TRR files, which were then sorted by the publisher. For the search, we used reputable online resources such as Google, Springer, Elsevier, and similar databases. We compiled a research database of abstracts, keywords, and titles and carefully analysed both the content and algorithms of each publication. In this way, we were able to classify the publications on the topic of TRR. In this research, we employed the search strings "Test Redundancy Reduction," "Test Suite Reduction," and "Test Case Reduction."

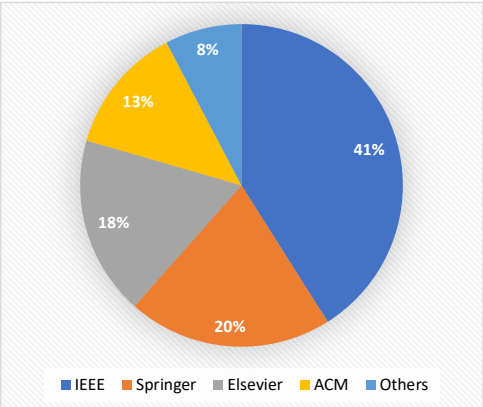

Fig. 1. Percentage of journal-specific articles on TRR

2.3 Inclusion and Exclusion Criteria

2.3.1 Inclusion criteria

i. Research publications satisfy the search string.
ii. All Papers that are written in English.
iii. Papers published in a conference, journal, and book chapters.
iv. Paper that has full-text available

2.3.2 Inclusion criteria

i. The title, abstract, or even their content was not closely related to our search string, however without any semantic interplay.
ii. Personal blogs, webpages, and PowerPoint presentations.
iii. No similarity with the research theme, or even the focal aim was completely contrary to the purpose of the issues addressed in the RQs.
iv. Irrelevant and Gray studies
v. Papers are not in full-text form.

3. Taxonomy of TRR Methods (RQ#1)

In the context of the TRR problem, four main categories of techniques have been identified: exact, heuristic, meta-heuristic, and machine learning. Further subcategories have been established within each of these four groups, as illustrated in Figure 2. This taxonomy serves as a comprehensive framework for future research on TRR, providing a clear and organized system for the classification and comparison of different techniques. By breaking down the main categories into subgroups, researchers will be better equipped to identify the most appropriate techniques for their specific needs and to assess the effectiveness of various techniques in the TRR problem. This taxonomy will therefore contribute to the ongoing development and refinement of TRR techniques, facilitating progress in this important area of research.

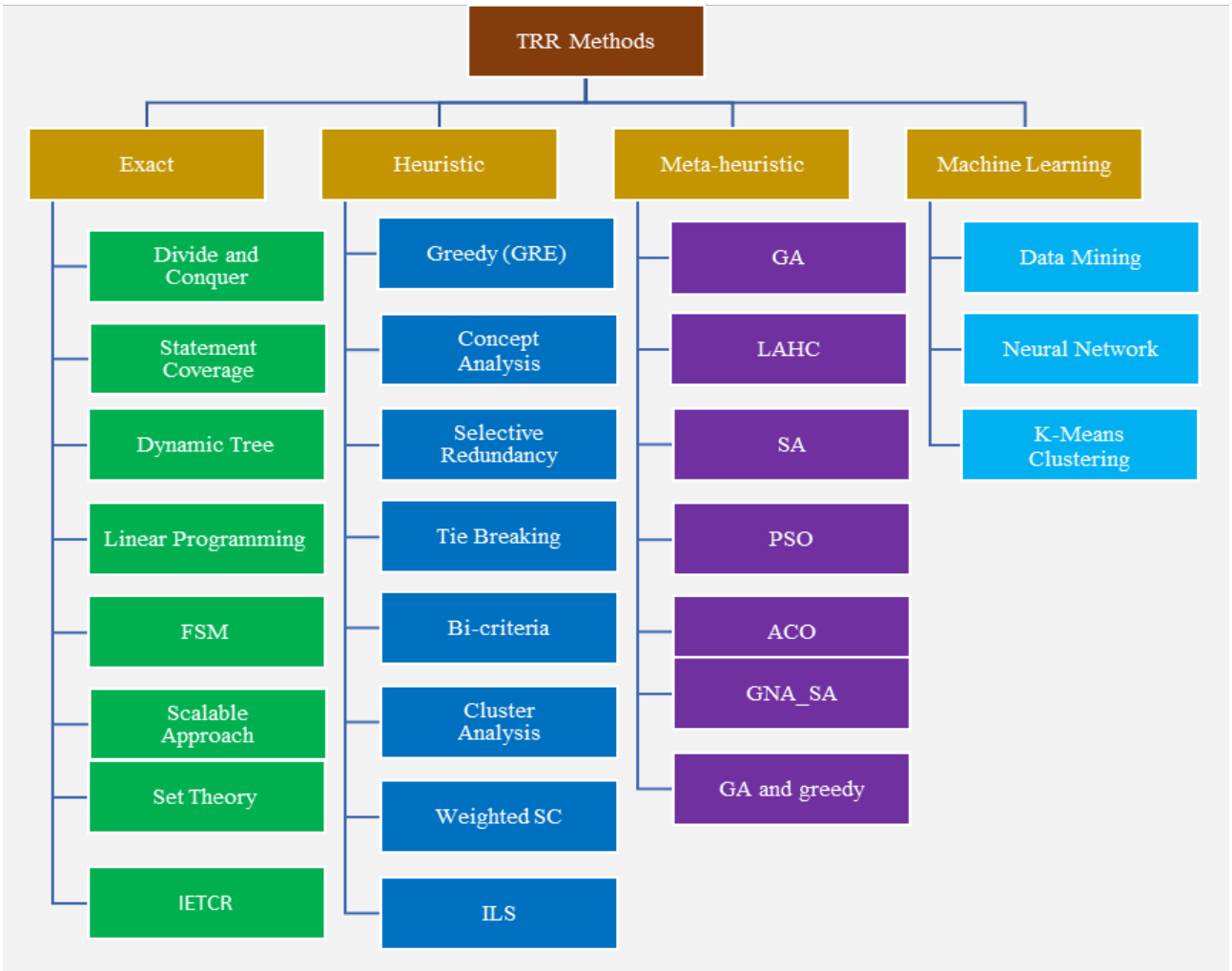

Fig. 2. Taxonomy of TRR methods used in various research articles

3.1. A Brief Review of Research Articles on TRR Adopt Exact Methods (RQ#2)

Combinatorial optimization problems of finite size can always be solved by exact algorithms, which guarantee the identification of an optimal solution and its optimality confirmation in a finite and instance-dependent runtime. However, in practice, it is not always feasible to compute optimal solutions efficiently. Therefore, one must choose between optimality and efficiency. In other words, it may be necessary to compromise on the guarantee of finding optimal answers in Favor of obtaining high-quality solutions within a polynomial time frame [8]. A brief review of research articles on TRR that adopt exact methods is given as follows:

Chen and Lau [9] conducted a study that investigated the characteristics of representative sets in the context of the divide-and-conquer strategy for reducing test suites. They concentrate on partitioning methods that preserve the minimum and good narrative sets in terms of critical test cases. Proposed strategies involve breaking down the main issue into smaller sub-issues, determining the best solutions to each sub-problem, and then using these outcomes to build a solution to the original problem. They took the necessary cases for testing and discarded the unnecessary ones, yielding an important and redundant set of tests. An essential test case is a member of the essential subset, while a subset that can satisfy its met requirements through additional test cases is deemed redundant. The essential subset can be utilized to create a representative set, while the redundant subset can be eliminated.

Khan and Nadeem *et al.,* [10] proposed a Test Filter that employed the statement-coverage criterion to reduce test cases. The approach appoints weights to each test case that correspond to the number of instances in which the test case addresses various statements. Based on these weights, the method selected non-redundant test cases that achieved complete statement coverage. The test cases were then arranged in a weighted set, and those with higher weights were selected first. Once all the requirements were satisfied, the Test Filter chose test cases with lower weights. If there was a tie among the test scenarios, a random choice was selected. A reduced set of test cases was obtained by applying the Test Filter, and a few test scenarios were added to expand the reduced set. To test the effectiveness of the Test Filter, they conducted an experiment on the triangle problem. The results indicated that the represented method efficiently minimized the test suite's size by almost 90%, while also lowering the associated costs, such as those associated with execution, storage, and management. Moreover, the approach required less storage space and CPU time for the selection of test instances.

Smith *et al.,* [11] proposed a novel approach for prioritization and reduction of test cases using call trees. Their approach involved creating a model of program behaviour based on trees and reordering the test cases by priority to meet all the deadlines. The dynamic call tree was utilized to identify the set of tests that replicate certain paths across the tree, and the prioritization was based on coverage efficacy. They created a call tree function Object () {[native code]} using the Java 1.5 and AspectJ 1.5 programming languages, which initializes the call tree before running the first test case. Each time a test case is invoked, a node is added to the tool's call tree, and a distinct test requirement is created as each path represents a sequence of method calls that occurred during testing. They produced a minimal test suite that met all specifications but had fewer test cases than the original test suite using a reduction algorithm. Their research was conducted on a GradeBook program that has 147 methods, 10 classes, and 1455 non-commented source statements (NCSS). The investigation revealed that call tree construction probes increased the test set fulfilment time by 12.3%. By reducing the testing time by 82% while utilizing the overlap-aware greedy method, they were able to produce a smaller set with better coverage. The main test set's coverage effectiveness was 38, while the prioritized test set was 96. Although the construction of call trees took longer, they found that prioritized suites could attain coverage more quickly. Their condensed set had 45% fewer test cases than the original suite, providing better coverage effectiveness while reducing testing time and costs associated with test execution, storage, and management.

Fraser and Wotawa [12] proposed a model checker-based technique for reducing the size of a test set in terms of both the amount of test cases and the overall length of all test cases. The primary goal of the method is to find and remove redundant test cases from the suite. The proposed technique has an advantage over older methods as Under certain conditions, it has no detrimental impact on the quality of the resulting test suites in terms of test coverage or fault detection ability. To exhibit the effectiveness of the presented technique, the empirical results were guided, and the results showed a minor improvement in the quality of the resultant test suites. Although the reduction achieved by the technique is not as high as the methods that heuristically exclude test cases, its significance is confirmed by the trials. The technique can be used in conjunction with other approaches to achieve better results.

Chen, Zhang, and Xu [13] presented an improved ILP strategy for TSR that aimed to bridge the gap between heuristic and ILP strategies. The authors introduced a lower bound for the reduced test suite and aimed to find solutions close to the lower bound. If the representative set size equals the lower bound, the result is the best; if it is closer to the lower bound, the result is also acceptable; if it is farther from the lower bound, ILP or other valuable techniques must be utilized to improve the representative set. To define that there are no 1-1 extra test cases or requirements, the authors first

applied a 1-1 reduction to test cases and requirements. Their approach was based on the single-branch concept, where the most viable subproblem for each variable is chosen. The authors competed for their approach with other alternatives and discovered that their method consistently outperformed them and occasionally guaranteed a minimal size-reduced set. DILP was able to produce a minimal test set for the rest of the Boolean criteria. nevertheless, the authors noted that their approach was more sophisticated than heuristic techniques.

Miao *et al.,* [14] present a novel method for eliminating repetition in test sequences using string matching. The authors also define reduction rules for the four distinct types of redundancy found in test sequences. To ensure the efficiency of redundancy reduction, a transformation rule that converts ineffective test segments into effective test sequences is suggested. Based on these rules, a revolutionary algorithm is proposed and implemented using Java. Case studies have shown that their method is beneficial. Compared to previous studies on redundancy reduction, this method has a superior ability to remove redundant test sequences while encouraging the use of finite state machine (FSM) based test coverage requirements in real-world testing scenarios. This method has great potential for enhancing the efficiency of testing while minimizing the cost and time required for testing.

Galeebathullah and Indumathi [15] presented a TSR technique based on set theory (ST). They utilized ST and the greedy approach to create minimized sets of test cases. To identify specific requirements that have not been covered, the intersection function was employed. The intersection of the one-branch coverage criterion requirement and other branch coverage criterion requirements for the collection of test cases was determined using set theory. The intersection of requirements was established first, and the test case was included in the smaller set of cases if any intersectional elements appeared. This process was repeated until all conditions were met. To evaluate their method, they tested it on a little software application based on branch coverage requirements. Their method produced test suites that were the same size as those created by conventional methods and covered all the requirements. The proposed set theory technique is promising as it lowers the number of tests without sacrificing coverage. as the conventional methods. However, it needs to be tested on increased programmed size and complexity to validate its efficiency and effectiveness.

Chen *et al.,* [16] propose an approach for the automatic creation of test cases using Input/Output Symbolic Transition Systems (IOSTS). These systems are commonly used to model reactive systems with data. However, selecting test cases for a produced test suite based on test goals given by IOSTS or historical logic can lead to redundancy. In order to reduce the cost of testing implementation, various techniques for removing redundancy are presented. IOSTS are the test cases where these methods are used. This approach has the advantage of reducing the cost of developing test cases and the size of the test suite. By demonstrating the effectiveness of the proposed technique in a case study, the authors show that the approach is practical and feasible. In addition, the use of IOSTS ensures that the generated test cases are effective and reliable in meeting the required test goals.

Nasir *et al.,* [17] proposed the use of entropy for detecting and eliminating duplicated test cases generated from the Control Flow Graph (CFG). They demonstrated that their strategy was able to eliminate 61% of test cases compared to the initial test suite. Entropy was used as a novel approach for detecting and removing duplicate test cases, offering a new strategy for test case minimization. The proposed method computes the entropy values for each unique path in the CFG and removes test cases with duplicate entropy values. The use of entropy as a metric allows for a more granular and effective approach to test case reduction. The study includes an in-depth analysis of the suggested procedure and the outcomes of their experiments, demonstrating the efficacy of the approach.

Marijan and Sen [18] present a novel approach to detecting and reducing redundancy in test suites for highly configurable software. The authors utilize equivalence partitioning to identify test cases that cover the same sets of feature interactions. Following this, they identify partitions that contain multiple test cases and exclude those that do not have distinct partitions. The methodology is evaluated through an industry case study and several test suites that were created as a result of the case study. The results of the study indicate that their approach has the potential to reduce test execution time by an average of 35% compared to standard industry practice, with no discernible impact on fault detection efficiency. The paper presents a step-by-step guide on how to apply their methodology to detect and reduce redundant test cases. The authors suggest that their approach could be used to identify redundant test cases in other highly configurable software, thereby improving efficiency and reducing the time and cost associated with testing.

Özener and Sözer [19] proposed a new approach to multi-criteria TSR by formulating the problem as an ILP model. They discovered shortcomings in the previous ILP models that could produce suboptimal results. To overcome these issues, the authors introduced opposing instances to illustrate how their formulation can address the limitations. The effectiveness of their approach was empirically evaluated using a publicly available dataset from various open-source projects. The results demonstrated that the proposed linear formulation outperformed state-of-the-art models in terms of the identical objective function and sets of criteria such as statement coverage, fault-revealing capabilities, and test execution time. Furthermore, the linear model had better time performance than non-linear models, allowing it to be used in more complicated situations. Overall, this approach offers an improved method for multi-criteria TSR.

A method called Schema Test Integrity Constraints Combination for Efficient Reduction (STICCER) has been proposed by Alsharif and Kapfhammer [20] to systematically merge and discard redundant tests to create a smaller test suite. STICCER is different from traditional methods like Greedy and HGS, as it minimizes both the overrun in test criteria coverage and the database state produced by the tests. Authors assert that STICCER represents a significant step forward in TSR. To validate their claims, the study compared the effectiveness of STICCER with Greedy, HGS, and a Random methodology, using 34 relational database schemas and test data generated by two test-generating methods. The results suggest that STICCER is more effective than the other methods, as it provides the same coverage as the original test suite while creating a smaller test suite with reduced overlap and database state.

Wang and Du [21] presented a novel approach for reducing the number of test cases, referred to as the information entropy-based test case reduction (IETCR) approach, which aims to localize mutation-based faults. The IETCR approach employs information theory to compute the entropy change of the test cases. The test cases are sorted in ascending order based on their entropy values, which are then used to produce a rank list of the test cases. The IETCR approach considers both successful and unsuccessful test cases and retains all unsuccessful test cases while choosing a few successful test cases according to the original mutation-based fault localization (MBFL). To evaluate the IETCR approach, the authors conducted experiments on 112 faulty versions from six real-world programs. The evaluation results demonstrate that the IETCR technique performs better than two state-of-the-art baseline procedures in respect of fault localization accuracy.

Table 2 presents a summary of research articles on TRR that have utilized exact methods. The table provides information on the methodology used in each study, along with the contribution identified by the authors. The remarks column in the table includes a brief description of their limitations and gaps, which can be used to guide future research in the field of TRR. For both academics and professionals, the table is a valuable resource for those who are interested in

implementing exact methods for TRR and want to stay up to date with the latest developments in the field.

Table 2
Summary of Research Articles on TRR that Adopt Exact Methods

Author(s) [Reference]	Year	Technique Adopted	Contribution	Remarks
Chen and M. F. Lau. [9]	2002	Divide and conquer method	TSR dividing techniques connected to this approach	The concepts of redundant test cases and redundant subsets may provide further insights for creating new, comprehensive splitting techniques for test suite reduction. More investigation in this area is worthwhile.
Khan, A. Nadeem *et al.,* [10]	2006	Statement Coverage	The ability to identify and eliminate unnecessary test cases	we provide our findings in the hopes of providing the test manager with a practical and economical method for cutting down on the amount of test cases. The effectiveness of the tests is ultimately increased.
Smith, J. Geiger. *et al.,* [11]	2007	Dynamic trees	Constructing call trees increases 13% testing time	To limit the size and runtime of the test suite, the reduction method chooses a subset of the original tests while still ensuring coverage of all the possible paths through the call tree.
Fraser and F. Wotawa. [12]	2007	Model-checker based approach	the total size of to test set is minimized	The algorithm's run-time complexity is one disadvantage of this method. Nonetheless, even without additional improvements, the method works flawlessly with actual test suites.
Chen, X. Zhang, *et al.,* [13]	2008	Liner Programming	The issue is complicated yet can be resolved in polynomial time.	To fill the gap between ILP and more conventional heuristic approaches, the degraded ILP (DILP) strategy was presented. In order to find a workable solution, DILP can generate a lower bound of minimum size.
Miao, P. Liu. *et al.,* [14]	2009	Finite state machine	Finite state machines	The study addresses test sequence redundancy from multiple classic test coverage requirements using a basic FSM. A transformation rule converts inefficient test segments into effective test sequences.
Galeebathullah. [15]	2010	Set theory	All conditions are met, and the reduced set is the same as greedy and HGS.	Set theory is a key idea in mathematics and computer technology. Every mathematical concept begins with sets. For instance, associations between two items are represented as a collection of ordered object pairs.
Chen, X. Li. *et al.,* [16]	2010	Model-based testing method	It has the potential to lower not just the size of the test suite, but also the cost of developing test cases.	To keep things straightforward, we're only looking at deterministic models; however, more complicated IOSTS elements like as recursion and concurrency should be addressed.
Nasir, N. Ibrahim *et al.,* [17]	2017	Entropy	When compared to the original test suite, the approach reduced 61% of the test cases.	This method has been validated using a straightforward sizing program and examined by hand.
D. Marijan and S. Sen. [18]	2017	Own approach	On average, our technique can reduce test execution time by 35%.	Our findings could indicate that the industrial data sets we used were not adequately represented. All implementations utilized in the experiment were properly evaluated and tested.

Özener and H. Sözer. [19]	2020	Liner Programming	Our method produces either superior or identical outcomes.	These compositions do not promise a rapid solution. As the problem expands in magnitude, this solution may fall short.
Alsharif, G. M. Kapfhammer *et al.,* [20]	2020	STICCER	The most successful method for minimizing the amount of test cases is STICCER.	It is impossible to assert that they are representative of all schemas, and it is also challenging to find a set that is appropriately representative. However, our collection of schemas comes from various sources and ranges widely in size and complexity.
Wang, B. Du *et al.,* [21]	2020	IETCR	The technique can decrease costs by 56.3% to 88.3% while maintaining about the same fault localization accuracy.	The technologies employed in our experiments to cause mutation present the most hazards. While utilizing various mutation methods may have an impact on our empirical findings

3.2. A Brief Review of Research Articles on TRR Adopt Heuristic Methods (RQ#2)

Heuristic algorithms are effective in solving certain problems but may not perform well in other domains due to being specialized in specific problems. The set covering problem was initially proposed by Chvatal [22] as a problem for which a heuristic algorithm could be used to solve. Although heuristic algorithms may be quick to solve issues in a limited area, they may not be equipped to handle highly complex optimization problems. The literature has many discussions on heuristic algorithms, which we have classified based on the focus keyword of the articles. A brief review of research articles on TRR that adopt heuristic methods is given as follows:

Harrold *et al.,* [23] introduced a heuristic algorithm for minimizing test cases, which involves selecting a narrative group of test cases that cover the identical ground as the total test set. The algorithm achieves this by recognizing and removing extra and old test cases. The delegate set can potentially result in a reduced test suite, which helps to identify the test cases that need to be rerun after modifying the software. In terms of testing methodology, the proposed approach is quite flexible, necessitating only a link between a testing specification and the test cases that satisfy the requirement. The information stream testing approach is employed to demonstrate the usefulness of the technique. Empirical evaluations demonstrate that significant reductions can be achieved using this approach. However, the technique may not be suitable for optimization problems with high complexity. Further research is needed to extend this technique for such problems.

Chen and Lau [24] proposed a GRE strategy for test case minimization that is based on three strategies: the Essential strategy, the 1-to-1 redundant strategy, and the Greedy strategy. The Essential strategy involves selecting the most important test cases and adding them to the illustrative group. The 1-to-1 redundant technique eliminates test cases that do not contribute to the coverage of additional requirements. To ensure that every requirement is met, the Greedy approach is then used for the other test cases. The GRE strategy is expected to produce the best narrative groups since it combines the strengths of the essential and greedy approaches. The approach has been evaluated on various software programs and has shown promising results. The authors also discuss the limitations and gaps of the GRE strategy, along with future research directions.

Jones and Harrold [25] introduce novel techniques for test-suite prioritization and reduction that can effectively utilize MC/DC principles. Far from previous approaches, their algorithms consider the intricacies of MC/DC when deciding whether to order or reduce a test suite. In the paper, we present

a case study that analyses how one program and several test suites fared after being subjected to the TSR technique. ReduceSuite is their proposed technique that selects the necessary test cases first and then removes the remaining extra test cases in the test sets using a series of steps. Firstly, the algorithm removes the test case that was not designated as essential and repeatedly identifies the weakest test case among those that have not been designated as essential. A weak test case is one that has a negligible impact on meeting the test case requirements. The reduced test suite is returned as a set of essential test cases, and the procedure terminates when those cases satisfy all of the original test suite's requirements. The proposed technique improves test efficiency and reduces the time and cost involved in testing while maintaining the same level of test coverage.

Tallam and Gupta's [26] introduced a Formal Concept Analysis-based approach for reducing the test suite size using a greedy-inspired technique. Formal Concept Analysis is a technique used for objects with discrete attributes. In this approach, test cases are recognized as objects, while their specifications are considered as attributes. The relationship between the object and its attributes is linked to the test case's coverage information. The Formal Concept Analysis framework enables the study of context, where the most comprehensive collection of objects and attributes, or contexts, can be identified object reduction rules, and attribute reduction rules are used to achieve this goal. Traditional greedy heuristics only employ object implications, whereas attribute implications are not considered. Therefore, Tallam and Gupta's approach proposes the use of attribute implications in the reduction process to improve the efficiency of the reduced test suite. The presented approach was tested on various software systems to evaluate its effectiveness in reducing the size of the test suite. The results demonstrate the efficacy of the approach in reducing the size of the test suite while maintaining the coverage of the original test suite.

Jeffrey and Gupta [27] proposed a method for reducing test suite size while minimizing the loss of fault detection ability. The primary and secondary factors for coverage were employed to accomplish this. Test cases covering the vast majority of missing needs were chosen based on the primary criteria, while redundant test cases were chosen based on the amount of def-use coverage they provided, in addition to meeting primary requirements. After each test case was added, the newly covered needs were noted and dropped from consideration, and the coverage information was updated. All possible test cases were picked in this way until either none remained, or it became clear that they did not fulfil any additional requirements. The proposed method was tested using Siemens programs, and it produced representative sets with improved defect detection efficiency.

Lin and Huang [28] explored test suite minimization by developing an improved tie-breaking algorithm. In contrast to conventional methods, where a random decision is made, they employed extra coverage criteria to break a tie between two test cases. Coverage information and def-use pairs were used as the first and second criteria, respectively, to break the tie between test cases. They used HGS and GRE methods and updated the 1-to-1 redundancy technique in the modified GRE (M-GRE) approach. In the modified HGS (M-HGS) technique, the test case that covers the most secondary needs is selected when two test cases are tied. To compare their results, they used Siemens suite data and Space data. They discovered that M-HGS and M-GRE generate minimized sets with higher defect capture capabilities when compared to the main HGS and main GRE. Their approach implies the effectiveness of defect discovery in the minimized test suites. Overall, their findings suggest that the improved tie-breaking algorithm can produce test suites with higher fault capture capabilities, leading to better defect identification efficiency.

Khalilian *et al*., [29] represented a bi-criteria TSR method based on the clustering experiment of perfection patterns. Distribution-based and coverage-based methodologies were used to build all feasible coverage-reduced test suites with minimal crossover between implementation profiles. The coverage-based approach utilizes the def-use pair criterion to choose test cases, covering possible

fault-containing execution routes. On the other hand, the distribution-based techniques use two strategies, cluster filtering and failure pursuit, which group test cases based on how they execute. In cluster filtering, test cases are divided into clusters where items with related features are placed in the same group. Then, test cases are randomly selected from each group to form a full coverage reduction set with minimal overlap. The suggested techniques were assessed on the Siemens suite. The empirical evaluation showed that their method can generate smaller test suites with lower effective defect identification, making it a promising approach to test suite reduction.

Parsa *et al*., [30] presented a novel cluster analysis-based approach for TSR that integrates coverage-based and distribution-based methods. The proposed method first determines the execution profile of each test case and then groups them based on their similarity. Test cases with similar execution patterns are assigned to the same cluster to ensure that the same program parts are covered by test cases from the same group. A heuristic approach is taken to select test cases from each cluster, ensuring that the coverage of the minimized test suite is equal to that of the main suite. They conducted an experiment on the Siemens suite, seeding a single fault into each program. The results showed that their approach produced a smaller test suite with acceptable coverage but was less effective in detecting faults compared to the H method. They used Weka 3.5.8 and the CLOPE clustering algorithm, which can handle large and high-dimensional data, and Repulsion, a variable that controls the degree of inter-cluster similarity. Adjusting Repulsion can change the number of clusters.

Xu *et al*., [31] proposed a weighted greedy algorithm known as Weighted Set Covering Techniques to reduce the size of test suites. Initially, the algorithm checks if there is already a test case that fulfils the overall specifications. If there is, the algorithm selects that test case. If not, it removes one-to-one duplicate test cases, improves the test suite, and adds any unmet criteria. The algorithm prioritizes the most important test cases and arranges them in descending order to fulfil the undiscovered requirements that are still present. The prioritized test cases are added to the smaller collection. The optimized test suite obtained using this method is more effective. To conduct an efficiency analysis of the suggested method, the authors conducted experiments using the real Navigation Model test set. The results demonstrated that the proposed method produced test suites with the smallest sizes and costs. The weighted greedy algorithm presents a promising solution for reducing the size and cost of test suites, which can improve the efficiency and effectiveness of software testing.

Xu *et al*., [32] proposed two heuristic algorithms based on the Harrold-Gupta-Soffa (HGS) algorithm: Non-Redundant HGS and Enhanced HGS. These algorithms are designed to generate smaller and non-redundant test suites. Non-redundant HGS selects a test case for larger values based on the total coverage of unmarked related testing suites. On the other hand, Enhanced HGS selects a test scenario for larger cardinalities based on the completeness of linked testing sets that have not been tagged. Experimental studies show that these algorithms can invariably choose reduced-size test suites in comparison to the current HGS heuristics. Moreover, these algorithms also enhance the efficiency of the testing process by reducing the number of redundant test cases while ensuring that the testing coverage is maintained. These algorithms have been successfully applied to various software systems, demonstrating their practical usefulness in reducing the cost and effort of software testing.

Cruciani *et al.,* [33] proposed a unique class of methods for test suite reduction based on similarity by combining algorithms from the big data domain with heuristics to select a balanced sample of test cases. The techniques take the test cases themselves as input, making them widely applicable to test source code or command line input. Four strategies are compared in two versions: one that holds a budget B of test cases and another that reduces the test suite while ensuring some defined coverage.

The experiment demonstrates that the proposed methods produce a defect identification loss similar to current methods when giving significant efficiency advantages. The more effective of the four algorithms could pick B test cases in less than 10 seconds when used on a set of over 500K actual test cases. These methods give an outstanding outcome to the scalability challenge combated by many testing teams.

Mohapatra *et al.,* [34] have proposed an algorithm called intelligent local search (STAGE) for minimizing the size of the test suite. Test case minimization has been carried out in the past using various strategies, but the pursuit of an efficient strategy is still ongoing due to the problem's intrinsic complexity. The proposed method uses hill climbing for local search, which has been restarted several times. However, the restart points for local searches are not chosen randomly. Instead, careful choices are made to select the new starting point. After using this strategy, encouraging results were observed for the selected programs. The approach taken in this paper is unique and innovative and adds to the existing body of knowledge about test suite minimization.

Table 3 presents a summary of research articles on TRR that have utilized heuristic methods. The table provides information on the methodology used in each study, along with the contribution identified by the authors. The remarks column in the table includes a brief description of their limitations and gaps, which can be used to guide future research in the field of TRR. For both academics and professionals, the table is a valuable resource for those who are interested in implementing heuristic methods for TRR and want to stay up to date with the latest developments in the field.

Table 3
Summary of Research Articles on TRR that Adopt Heuristic Methods

Author(s) [Reference]	Year	Technique Adopted	Contribution	Remarks
Harrold, R. Gupta. *et al.,* [23]	1990	Heuristic H	Created a set of lesser size	As long as the relationship between requirements and test cases can be established, the process is independent of the testing methodology employed.
Chen and M. F. Lau. [24]	1998	Heuristic GRE	An optimum representative subset was created	One unique aspect of GRE is that even if the greedy strategy has never been used, we may still determine that the representative set is optimal.
Jones and M. J. Harrold. [25]	2003	Break-down and build-up TSR approach	The reduced test suites produced by the break-down technique are consistently less than those generated by the build-up approach.	Our method is flexible enough to be used for either single- or multiple-entity criteria. AllTests are the test cases in the test suite annotated with the conditions they cover; allConditions are the conditions in the program under test that are used as inputs to ReduceSuite.
Tallam Sriraman and Gupta Neelam. [26]	2005	Inspired Greedy Algorithm	All reduced sets were smaller than or equal to those generated by the greedy method.	it uses the connections between test cases and the connections between coverage requirements, which were only used separately in the previous work.
Jeffrey and N. Gupta. [27]	2005	Selective redundancy	An increase in the size of the test suites without a corresponding decrease in their ability to identify errors	The key distinction is that while minimization techniques aim to minimize redundancy, our novel algorithm explicitly aims to include redundancy in the reduced suites.

Lin and C. Y. Huang. [28]	2009	Enhanced tie-breaking techniques	Increased efficacy of defect detection	In the course of the reduction procedure, ties were broken using an extra coverage criterion. The outcomes might imply that the suggested strategy can improve the test case selection.
Khalilian and S. Parsa. [29]	2009	cluster analysis-based bi-criteria technique	Fewer test suites with improved problem-detection	For software testing research, Siemens's programs are standard fare. However, these tools have their limitations and flaws are well-known.
Parsa, A. Khalilian, *et al.,* [30]	2009	A new algorithm based on Cluster Analysis	The smaller suite provides appropriate coverage	The percentage of fault loss is calculated based on a simple cost model that considers all faults to be of equal significance. But, in reality, flaws might vary greatly in severity.
Xu, H. Miao, *et al.,* [31]	2012	Weighted Set Covering Techniques	Reduced test suite size and reduced cost	The optimization of large-scale test suites is an area where our method still needs additional research.
Gladston, HK Nehemiah. [32]	2015	HGS algorithm	Choose a shorter test suite size than the previous HGS heuristics.	In order to eliminate redundant test cases from the reduced test suite, they merged the technique of GRE into HGS, known as Non-Redundant HGS.
Cruciani, B. Miranda, R. *et al.,* [33]	2019	Scalable approach	Enhanced heuristic approach and a proposed new approach	The proposed methodologies are more efficient in terms of reduction time than all of the compared approaches except GA applied to function coverage, even for relatively small-scale benchmarks.
Mohapatra, A. K. Mishra, *et al.,* [34]	2020	Local search	Can provide very good representative sets in a short period of time	they frequently encounter issues with severely confined situations where potential solution spaces are not connected.

3.3. A Brief Review of Research Articles on TRR Adopt Meta-Heuristic Methods (RQ#2)

Meta-heuristic algorithms have become increasingly popular due to their capability to handle complex and large-scale computing tasks. These algorithms are universal, they can be used to solve a variety of issues and provide satisfactory solutions [54]. One of the most common applications of meta-heuristic methods is for solving NP-hard optimization problems, including TRR. TRR is an NP-complete problem due to a large number of potential solutions, and it requires extensive computational effort for evaluation. Therefore, meta-heuristic approaches have been widely used to efficiently solve TRR problems. A brief review of research articles on TRR that adopt meta-heuristic methods is given as follows:

Nachiyappan *et al.,* [35] presented a GA approach for TSR, which employed a mathematical model to minimize the number of tests. They began with a mathematical model that was used to generate the initial population determined by the test data. The technique then calculated the fitness value of the test cases using the block-based coverage and implementation time. The test cases with the highest fitness value were then selected while filtering out those that violated fitness constraints. Through this strategy, the test suite size was reduced while maintaining the same level of coverage. The proposed GA approach provides a systematic way to determine an optimized set of test cases and is an efficient method to resolve the NP-complete issue of test suite reduction. The effectiveness of the presented algorithm is validated by empirical results, which illustrate that it can reduce the test suite size by up to 45% while retaining the same level of coverage. This approach is promising in its ability to provide cost-effective and time-saving solutions for software testing.

Zhang *et al.,* [36] proposed an Improved Quantum Genetic Algorithm (IQGA) for TSR. The algorithm employs quantum bits (qubits) as information bits to encode chromosomes, and the vector

description of the quantum serves as the basis for IQGA. The chromosome, represented by qubits, is updated through a quantum rotating gate, which can be dynamically adjusted according to the fitness value of each individual. This method significantly reduces the complexity of the query computation and simplifies it. The authors have compared their approach with conventional methods and demonstrated that it dramatically lowers testing expenses while increasing test efficiency. The compressed test set generated by IQGA is much smaller than that generated by conventional methods, resulting in greater efficiency.

You and Lu [37] proposed a GA for the Reduction of the Time-Aware Regression Testing issue. The study proposed a new genetic method for the reduction problem in time-aware regression testing. It delves into the parent selection, crossover, and mutation operators of the genetic algorithm, as well as the representation and fitness function of the GA. The proposed algorithm reduces the overall execution duration of the remaining test cases and eliminates extra test cases from the regression testing sets. The experiment was conducted on eight example programs to test the effectiveness of the GA. The results demonstrate the efficiency of the suggested GA in mitigating the difficulty of time-aware regression testing.

Bakar and Zamli [38] proposed a novel approach to minimize test redundancy and prioritize tests based on the Late Acceptance Hill Climbing (LAHCS) algorithm. The authors utilized LAHCS as a research tool to assess the efficiency of LAHCS in reducing test redundancy and prioritizing test cases. One of the notable strengths of LAHCS is its simplicity, as it has only one parameter for modification and is not susceptible to poor parameterization and tuning. The strategy employed by LAHCS is to prioritize test cases before beginning the search process. The authors compared the performance of LAHCS with three benchmark techniques, namely GRE, GE, and HGS, and observed comparable scores for LAHCS. The experimental outcomes validated the efficacy of the suggested method in minimizing test redundancy and prioritizing tests.

Zamli *et al.,* [39] developed tReductSA, a novel approach for reducing test cases in a systematic manner. The proposed approach utilized a Simulated Annealing-based optimization algorithm and a methodical merging strategy to optimize the selection of test cases for removal. The approach was benchmarked against other works, such as GE, GRE, and HGS, and showed a comparable level of optimality while providing a wider range of solutions. The proposed approach has the potential to minimize the cost and effort of testing while ensuring the efficiency and effectiveness of the testing process. The results of the benchmark tests highlight the efficiency of tReductSA and its ability to scale well with other related works. Overall, this approach can be a valuable addition to the field of software testing, allowing for more efficient and effective testing while maintaining the quality and accuracy of results.

Mohanty *et al*., [40] introduced an ant colony optimization-based approach for test set reduction. Their approach shows the total test number in a complete graph as nodes, with a matrix holding the execution time for each test case and its associated test requirement. The ants start at nodes of the complete graph, and they choose neighbouring nodes based on maximizing coverage requirements and minimizing execution time using the matrix. The approach discovers a narrative set of test cases that fulfil whole criteria and require minimal processing time. In their evaluation, Mohanty and Mohapatra demonstrated the effectiveness of their algorithm by comparing the size of the RS produced by their approach with the size produced by other test suite minimization techniques. They showed that their approach generates a much smaller RS while providing adequate coverage.

Coviello and Romano [41] proposed GASSER (Genetic Algorithm for Test Suite Reduction), a new approach for Test Suite Reduction (TSR) that takes into account statement coverage, test-case variety, and TS size. They proposed seven different GASSER cases, each using several metrics to measure test-case diversity. In order to evaluate the effectiveness of GASSER, an experiment was

driven comparing it to nine baseline methods in terms of the reduction in TS size and the elimination of the fault-detection ability of the reduced TS. The experiment demonstrated that some instances of GASSER significantly reduced the amount of the TSs with only a minor impact on fault-finding efficiency compared to the baseline methods. These findings suggest that GASSER is a promising approach for TS reduction that can effectively balance the trade-off between test coverage and size reduction.

Xia and Zhang [42] present a novel evolutionary multi-objective optimization approach for reducing the cluster test suite. Specifically, the authors use the K-means technique to aggregate related test cases into a cluster. The clustering results are then used by a multi-objective evolutionary approach to eliminate extra test cases based on coverage-related criteria, fault-related criteria, and cost-related criteria. The proposed technique was tested on eight subject programs, and the result findings show that it performs well in three state-of-the-art methods in terms of fault detection (4.61% to 9.44%) and reduction ratio (4.10% to 10.64%). Furthermore, the expenses associated with the proposed method are comparable to those of the other methods. The results of this study suggest that the proposed multi-objective optimization approach is an effective method for reducing the cluster test suite while maintaining fault detection rates and cost efficiency.

A discrete combinatorial gravitational search algorithm (DCGSA) was presented by Bajaj and Sangwan [43] for test case prioritization and minimization problems. The DCGSA algorithm includes a fix-up mechanism that perturbs the population of each iteration. This approach is applied to several subject programs of varying sizes, and the outcomes indicate the efficiency of the suggested approach for all programs. Further, the updated GSA has the most compressed dispensations, demonstrating the algorithm's robustness in boxplots. The interval charts display that the presented approach needs the fewest test cases to guarantee complete coverage. The proposed algorithm's performance is also compared to state-of-the-art methods, and the results show that the DCGSA algorithm is an effective approach to test case prioritization and minimization.

Deneke *et al.,* [44] have introduced a new approach to reducing the number of test cases in a suite using a PSO algorithm. Using metrics like reduced set size and runtime, the suggested method was compared to four other benchmark reduction strategies. The experimental findings on the identical input dataset show that the test suite's reduction percentage by PSO is 55.55%, by G WSC is 22.22%, and by G, HGS & GRE is 44.45%. In terms of fulfilment cost, G WSC, G& GRE, HGS, and PSO have values of 63, 72, 67, and 43, respectively. The results show that the PSO approach demonstrated a promising and superior result compared to previous techniques. Therefore, the proposed method can be a reliable alternative for reducing the number of test cases in a suite.

Table 4 presents a summary of research articles on TRR that have utilized meta-heuristic methods. The table provides information on the methodology used in each study, along with the contribution identified by the authors. The remarks column in the table includes a brief description of their limitations and gaps, which can be used to guide future research in the field of TRR. For both academics and professionals, the table is a valuable resource for those who are interested in implementing meta-heuristic methods for TRR and want to stay up to date with the latest developments in the field.

Table 4
Summary of Research Articles on TRR that Adopt Meta-heuristic Methods

Author(s) [Reference]	Year	Technique Adopted	Contribution	Remarks
Nachiyappan; A. Vimaladevi. [35]	2010	Genetic algorithm	Developed the smallest practicable test suite taking into consideration runtime and code coverage.	We propose an adaptive mutation strategy in which n-point mutation is performed on rejected individuals. This operation mutates the supplied bit only if it meets the mutation requirement.
Zhang, J. C. Liu *et al.,* [36]	2010	Improved Quantum Genetic Algorithm	Possibility of significantly lowering testing expenses while simultaneously increasing test throughput	It can lower the complexity of the query computation and dynamically alter the quantum rotating gates in accordance with each individual fitness value.
You and Y. Lu. [37]	2012	Genetic algorithm	Minimal test suite and execution time	It rectifies the problem that the heuristic methods had. It is able to eliminate all test cases in the regression testing suite that are redundant.
Bakar, K. Zamli, *et al.,* [38]	2014	Late Acceptance Hill Climbing algorithm	When it comes to the percentage of decrease, LAHCS offers outcomes that are sufficiently competitive.	LAHCS has proven that introducing an acceptance feature balances intensification and diversification to improve Hill Climbing's performance, which has been criticized for getting trapped in local minima/maxima.
Zamli, M. H. Mohd Hassin, *et al.,* [39]	2015	Simulated annealing	In comparison to previous work, tReductSA also provides a wider range of solutions.	It requires very slow cooling in order to impose layout regularities.
Mohanty, S. K. Mohapatra, *et al.,* [40]	2020	Ant Colony Optimization	The test case size is significantly reduced by the proposed approach.	When dealing with big amounts of data, it has several issues in terms of convergence speed and solution correctness.
Coviello, S. Romano, *et al.,* [41]	2020	Genetic Algorithm	When compared to conventional techniques, it reduces the TS size considerably while having a minimal impact on fault-detection capacity.	We anticipate that a TSR strategy aims to significantly reduce the size of the TS while maintaining the original TS's capacity to detect faults.
Test Suite Reduction via Evolutionary Clustering. [42]	2021	Evolutionary Algorithm	The costs of the suggested method are found to be comparable to those of the other methods.	The experiments demonstrate, in addition, that our solution has the maximum percentage of code coverage and the lowest missing fault rate of the test case.
Xia, Y. Zhang, and Z. Hui. [43]	2021	Gravitational Search Algorithms	The proposed algorithms outperform the GA in terms of perfectness and effectiveness.	Its disadvantages include a slow convergence speed and a tendency to stall in local minima in the final few rounds.
Deneke, B. Gizachew Assefa, *et al.,* [44]	2022	Particle Swarm Optimization	When compared to previous strategies, our approach yielded a more promising and superior result.	It has a poor convergence rate during the iterative process, and it is prone to reaching a local optimum in high-dimensional spaces.

In order to solve the test case reduction (TRR) problem, hybrid meta-heuristic methods can be used, employing a variety of strategies. A brief review of research articles that have used hybrid methods to address the TRR problem is provided as follows:

Yoo and Harman [45] introduced a multi-objective approach for reducing test suites. A hybrid algorithm was proposed, which combines a genetic algorithm with an effective approximation of the greedy approach to achieve superior Pareto fronts. The main focus was on accomplishing a wide range of goals. The test criteria were mathematically described using objective functions. Both computational cost and statement coverage were taken into account as optimization criteria in a two-objective setting. They used a cost-cognizant variant of the extra greedy method. Three-objective optimizations used the traditional weighted-sum method, where the weighted total of code coverage per unit of time and fault coverage per unit of time were used to integrate code coverage, error recovery, and time complexity. Their proposed approach produced more effective testing decisions. The effectiveness of their approach was evaluated based on the comparison of their proposed method with other new techniques. The evaluations show that their method significantly outperformed other multi-objective approaches in terms of the amount of test cases chosen, code coverage, and fault coverage.

Zamli *et al.,* [46] introduced a hybrid meta-heuristic technique by integrating a GNA with SA to fix the test redundancy reduction issue. Their study, GNA_SA was utilized as a case study. The experimental findings demonstrate that the GNA_SA approach provides a superior reduction as compared to the original GNA and several other related works. This approach uses a combination of GNA and SA for optimization. The GNA is employed to explore the search space, while SA is used to enhance the algorithm's ability to escape from local optima. The authors evaluated the GNA_SA algorithm using various testing criteria, including the number of redundant test cases, fault detection rate, statement coverage, and time complexity The empirical results suggest that the suggested hybrid approach effectively reduces redundancy in the test suite while maintaining the same level of fault detection rate and statement coverage as the original suite.

Table 5 presents a summary of research articles on TRR that have utilized hybrid methods. The table provides information on the methodology used in each study, along with the contribution identified by the authors. The remarks column in the table includes a brief description of their limitations and gaps, which can be used to guide future research in the field of TRR. For both academics and professionals, the table is a valuable resource for those who are interested.

Table 5
Summary of Research Articles on TRR that Adopt Hybrid Methods

Author(s) [Reference]	Year	Technique Adopted	Contribution	Remarks
Yoo and M. Harman. [45]	2009	Genetic algorithm with a greedy approach	Decision-making for testing that is more efficient	The representativeness of the topics covered in this study is the main issue. Only more studies utilizing a larger variety of software artifacts and optimization strategies will be able to counter this issue.
Zamli, N. Safieny, and F. Din. [46]	2018	GNA and SA	Superior reduction to the main GNA and other activities.	Their hybrid approach has a problem with population diversity.

3.4. A Brief Review of Research Articles on TRR Adopt Machine Learning Approach (RQ#2)

In recent years, learning-based algorithms have gained widespread popularity due to their ability to make decisions similar to humans. Various types of learning-based algorithms, such as Deep

Learning and Machine Learning algorithms, are being developed to address different problems. This section, a brief review of research articles on TRR that adopt machine learning methods are given as follows:

Hooda [47] proposed a new approach for generating test cases that incorporates a UML state chart diagram and tables, as well as an artificial neural network (ANN) as an optimization tool for minimizing duplicates in test cases generated using the GA. The UML state chart diagram and tables are used for visualizing the test case development process. The backpropagation approach is employed to train an ANN by applying a series of test cases to the system's main version. The use of ANN helps in reducing redundancy in test case development. The proposed approach deals with the redundancy of generated test cases while also delivering optimal efficiency and code coverage. The findings indicate that the algorithm is effective in achieving its objectives.

Chetouane *et al.,* [48] proposed a machine learning-based approach for TSR, which integrates binary search and k-means clustering. The goal of the algorithm is to group together similar test cases and select one representative test case from each cluster to add to a more compact test suite. In order to minimize the test suite, the algorithm needs to determine the appropriate number of clusters that do not significantly deviate from the coverage or mutation score obtained from the original test suite. The proposed approach is presented alongside empirical outcomes obtained from a range of Java programs with varying input and output formats, ranging from small to large. The findings demonstrate an enormous drop in the sample sizes required for each test case when contrasted to other TSR strategies. The approach's effectiveness in minimizing the amount of the test set while retaining its efficiency is demonstrated by the experimental results.

Table 6 presents a summary of research articles on TRR that have utilized machine learning methods. The table provides information on the methodology used in each study, along with the contribution identified by the authors. The remarks column in the table includes a brief description of their limitations and gaps, which can be used to guide future research in the field of TRR. For both academics and professionals, the table is a valuable resource for those who are interested in implementing machine learning methods for TRR and want to stay up to date with the latest developments in the field.

Table 6
Summary of Research Articles on TRR that Adopt Machine-learning based Approach

Author(s) [Reference]	Year	Technique Adopted	Contribution	Remarks
Hooda. [47]	2018	Genetic algorithm and neural network	Maintaining maximum productivity and code coverage while minimizing unnecessary repetition in test cases	Complexity tends to overwhelm genetic algorithms. One major drawback of neural networks is their opaqueness.
Chetouane, F. Wotawa, *et al.,* [48]	2020	k-means clustering	Capable of significantly reducing the amount of test instances while needing a minimum reduction time	It is challenging for k-means to cluster data with heterogeneous cluster sizes and densities.

3.5. TRR Used in Different Comparative Studies

There has been some comparative research on numerous techniques in the literature of test redundancy reduction. These methods are as follows:

This study [49] compares four particular TSR techniques which are heuristic H, heuristic GRE GA, and ILP. The purpose of the study is to give suggestions for selecting the optimal test suite reduction approaches. Using the same platform, we developed four typical TSR methodologies and compared them experimentally using small and large subject applications. In their work, we discuss the findings

of our experiment and offer some analysis-based recommendations for choosing among test suite reduction approaches.

Zhong *et al.,* [50], a new experimental study of the four standard test set reduction approach is conducted. These techniques are the heuristic developed by Harrold *et al.,* the GRE heuristic by Chen and Lau, the genetic algorithm-based approach by Mansour and El-Fakin, and the ILP-based technique by Black *et al.,* Based on the outcomes of their empirical study, they give guidance for picking the most suitable test set minimization strategy and discuss some of the reasons behind the strategies varying degrees of success and efficiency.

Zhang *et al.,* [51] assessed an empirical investigation of Junit TSR. For Java applications with valid JUnit test suites, they applied TSR approaches. The usefulness of typical test-suite reduction strategies was investigated for larger applications. They suggested a course of action to the tester. They ran four real-world Java programmers through 19 iterations, each with its own set of manually seeded errors and JUnit test suites. They discovered that method H consistently provides nearly no decrease in the capacity to detect faults on both seeded and modified problems while obtaining the greatest reduction in test suite sizes. In real-world scenarios, Heuristic H should be employed to efficiently minimize costs. Techniques that significantly reduce the test suite likewise significantly reduce the capacity to detect mistakes.

Noemmer and Haas [52] apply four test suite minimization methodologies to distinct open-source software projects to analyse and access them. The fundamental greedy algorithm and the HGS algorithm were the two-statement coverage-based algorithms we employed. We examined the outcomes of full-suite mutation tests with the simplified test suites to determine how successfully a test suite's fault detection skills are preserved following test suite simplification. We discovered that there is a significant trade-off between decreasing the test suite and lowering fault detection capabilities using the methodologies we utilized. For each study subject, the amount of test cases was decreased by at least 50%; on average, our good algorithms eliminated 69% of the tests. Overall, test suite simplification has the potential to significantly reduce test execution time.

Rahman *et al.,* [53] focus on a comparative analysis of four metaheuristic algorithms (Teaching-Learning-Based Optimization, Jaya Algorithm, Sine-Cosine Algorithm, and Sparrow-Search Algorithm) applied to solve the test redundancy reduction problem. The study conducted multiple runs of each algorithm to ensure statistical significance. The results indicate that the Sparrow-Search Algorithm (SSA) is the most efficient metaheuristic algorithm for test redundancy reduction, considering both the average reduction rate and runtime effectiveness, compared to the other algorithms tested.

4. Discussion (RQ#3)

Many approaches to reducing test suites are presented in the literature, but there is confusion about which are most effective. For small projects, the exact algorithm provides optimal solutions, but it requires complex assumptions and a significant amount of time to run the program. It also requires a lot of memory and is not suitable for large problems. On the other hand, the heuristic algorithm provides a fast and simple solution for the design. However, it is not guaranteed to be optimal, it is problem-specific and cannot be used universally. Meta-heuristic algorithms can effectively overcome the drawbacks of exact and heuristic algorithms, and the solution time can be controlled. However, choosing an encoding and a fitness function can be difficult, and the convergence rate is slow. Although the Greedy algorithm-based technique can significantly reduce the number of test suites, it still requires optimization for large test suites. Hybrid approaches can reduce the size of test suites and simplify multi-objective optimization but at the cost of complexity.

Existing hybrid methods can be extended to include other methods. The ability to detect errors is reduced, but representative samples are obtained.

Nowadays, machine learning approaches are increasingly used to solve the TRR problem. This review provides an overview of the literature on TRR that uses machine learning approaches. Overall, it is found that machine learning techniques offer great potential for improving the efficiency and effectiveness of TRR. However, further studies and the development of more accurate and reliable TRR models are needed to improve the effectiveness of these methods. Figure 3 illustrates the percentage of methods used in the literature on TRR. Exact methods are the most commonly proposed by researchers which account for 33 percent. The heuristic and meta-heuristic methods both account for 32% each. Six percent of the total is accounted for by machine learning.

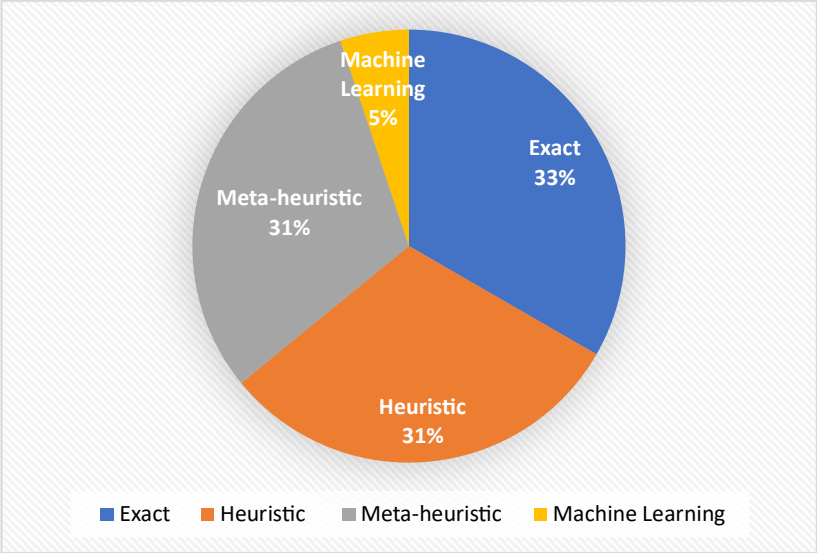

Fig. 3. Percentage of journal-specific articles on TRR

5. Future Research Direction

Many approaches to reducing test suites are presented in the literature, but there is confusion about which are most effective. For small projects, the exact algorithm provides optimal solutions, but it requires complex assumptions and a significant amount of time to run the program. It also requires a lot of memory and is not suitable for large problems. On the other hand, the heuristic algorithm provides a fast and simple solution for the design. However, it is not guaranteed to be optimal, it is problem specific and cannot be used universally. Meta-heuristic algorithms can effectively overcome the drawbacks of exact and heuristic algorithms, and the solution time can be controlled. However, choosing an encoding and a fitness function can be difficult, and the convergence rate is slow. Although the Greedy algorithm-based technique can significantly reduce the number of test suites, it still requires optimization for large test suites. Hybrid approaches can reduce the size of test suites and simplify multi-objective optimization, but at the cost of complexity. Existing hybrid methods can be extended to include other methods. The ability to detect errors is reduced, but representative samples are obtained.

i. Developing new machine learning algorithms for TRR problem solutions.
ii. Integrating multiple methods to achieve better results for TRR.
iii. Investigating the impact of using different evaluation metrics for TRR.
iv. Conducting empirical studies to compare the effectiveness of different TRR methods.

v. Develop domain-specific TRR methods to improve the efficiency of testing in specific domains.
vi. Exploring the potential of using TRR in conjunction with other software testing techniques such as fuzzing and model-based testing to improve the overall testing process.
vii. A High-level hybrid approach is absent for the TRR problem. So, it will be a good initiative for the TRR problem.

6. Threat to Validity

The validity of a literature mapping study can be compromised by many hazards. This study mitigated various potential limitations by incorporating established suggestions and standards for conducting literature mapping investigations.

i. One potential concern is that the research questions of this study may not encompass all facets of the current advancements in test redundancy research. In order to mitigate this potential risk, the researchers in this study employed the technique of brainstorming to establish a comprehensive set of research inquiries that encompassed the current body of knowledge in the field.
ii. There is no guarantee that all pertinent research on test redundancy reduction has been found. In order to find the pertinent research publications, several literature databases have been consulted, and a search string containing a variety of term synonyms (each paper's author proposed a different phrase that leads to the intended TRR concepts) has been employed. There might still be some unidentifiable papers, though. The snowballing technique was heavily used to address this problem and lessen the likelihood of overlooking significant linked studies.
iii. Application of the criterion may be hampered by personal prejudice and single-author decisions. To solve this problem, the authors came to a consensus before including or excluding any publication from the research.
iv. Data extraction may be negatively impacted by a single author occurrence. As a result, the data extraction procedure was carried out independently by each author, and the results were compared in an online meeting. All of the writers debated the discrepancies between the results during the discussion until a final, mutually acceptable compromise was established. Microsoft Excel's automatic filtering feature was also utilized to guarantee the precision of the data extraction procedure.

7. Conclusions

The current investigation has thoroughly examined the existing body of literature pertaining to various methodologies put forth with the aim of diminishing the quantity of test cases within the test suite. The methodologies under consideration encompass a range of sophisticated techniques, namely heuristic, meta-heuristic, exact, hybrid, and machine-learning methods. The findings of this comprehensive analysis suggest that every methodology possesses its own distinct array of merits and demerits, thereby rendering the task of ascertaining the most efficacious approach a formidable endeavour. Henceforth, this manuscript serves as a commendable compendium for scholars in pursuit of selecting a proficient methodology that harmonizes with the TRR framework. Nevertheless, it is imperative to acknowledge that a truly efficacious approach ought to encompass not solely the diminution of the test suite's magnitude, but also the astute identification and rectification of errors.

Hence, it is imperative to acknowledge that a methodology that demonstrates exceptional prowess in a singular domain may not prove to be satisfactory in its entirety. In its entirety, this review elucidates the present condition of the TRR quandary and serves as a catalyst for forthcoming scholarly investigations in this domain.

Acknowledgment
This research is supported by Fundamental Research Grant: A Multi-Factorial Agent Heroes and Cowards Algorithm based Strategy for Test Redundancy Reduction Problem (Ref:FRGS/1/2023/ICT02/UMP/01/1).

References

[1] Shin, Donghwan, and Doo-Hwan Bae. "A theoretical framework for understanding mutation-based testing methods." In *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 299-308. IEEE, 2016. https://doi.org/10.1109/ICST.2016.22

[2] Liu, Yang, Yafen Li, and Pu Wang. "Design and implementation of automatic generation of test cases based on model driven architecture." In *2010 Second International Conference on Information Technology and Computer Science*, pp. 344-347. IEEE, 2010. https://doi.org/10.1109/ITCS.2010.90

[3] Kumar, Gaurav, and Pradeep Kumar Bhatia. "Software testing optimization through test suite reduction using fuzzy clustering." *CSI transactions on ICT* 1 (2013): 253-260. https://doi.org/10.1007/s40012-013-0023-3

[4] Johnson, David S. "Approximation algorithms for combinatorial problems." In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pp. 38-49. 1973. https://doi.org/10.1145/800125.804034

[5] Singh, Rajvir, and Mamta Santosh. "Test case minimization techniques: a review." *International Journal of Engineering Research & Technology (IJERT)* 2, no. 12 (2013).

[6] Dalmia, Ritwik, Soumili Chandra, Sudhir Kumar Mohapatra, Srinivas Prasad, and Mesfin Abebe Haile. "A Systematic Literature Review on Test Case Minimization."

[7] Khan, Saif Ur Rehman, Sai Peck Lee, Nadeem Javaid, and Wadood Abdul. "A systematic review on test suite reduction: Approaches, experiment's quality evaluation, and guidelines." *IEEE Access* 6 (2018): 11816-11841. https://doi.org/10.1109/ACCESS.2018.2809600

[8] Dumitrescu, Irina, and Thomas Stützle. "Combinations of local search and exact algorithms." In *Workshops on Applications of Evolutionary Computation*, pp. 211-223. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. https://doi.org/10.1007/3-540-36605-9_20

[9] Chen, Tsong Yueh, and Man Fai Lau. "On the divide-and-conquer approach towards test suite reduction." *Information sciences* 152 (2003): 89-119. https://doi.org/10.1016/S0020-0255(03)00060-4

[10] Nadeem, Aamer, and Ali Awais. "TestFilter: a statement-coverage based test case reduction technique." In *2006 IEEE International Multitopic Conference*, pp. 275-280. IEEE, 2006.

[11] Smith, Adam M., Joshua Geiger, Gregory M. Kapfhammer, and Mary Lou Soffa. "Test suite reduction and prioritization with call trees." In *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, pp. 539-540. 2007. https://doi.org/10.1145/1321631.1321733

[12] Fraser, Gordon, and Franz Wotawa. "Redundancy based test-suite reduction." In *International Conference on Fundamental Approaches to Software Engineering*, pp. 291-305. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. https://doi.org/10.1007/978-3-540-71289-3_23

[13] Chen, Zhenyu, Xiaofang Zhang, and Baowen Xu. "A Degraded ILP Approach for Test Suite Reduction." In *SEKE*, pp. 494-499. 2008.

[14] Miao, Huaikou, Pan Liu, Jia Mei, and Hongwei Zeng. "A new approach to automated redundancy reduction for test sequences." In *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 93-98. IEEE, 2009. https://doi.org/10.1109/PRDC.2009.23

[15] Galeebathullah, B., and C. P. Indumathi. "A novel approach for controlling a size of a test suite with simple technique." *Int. J. Comput. Sci. Eng* 2, no. 3 (2010): 614-618.

[16] Chen, Donghuo, Xuandong Li, and Shizhong Zhao. "Auto-generation and redundancy reduction of test cases for reactive systems." In *2010 2nd International Conference on Software Technology and Engineering*, vol. 1, pp. V1-125. IEEE, 2010.

[17] Nasir, Noor Fardzilawati Md, Noraini Ibrahim, and Tutut Herawan. "Detection of Redundancy in CFG-Based Test Cases Using Entropy." In *Recent Advances on Soft Computing and Data Mining: The Second International*

Conference on Soft Computing and Data Mining (SCDM-2016), Bandung, Indonesia, August 18-20, 2016 Proceedings Second, pp. 244-252. Springer International Publishing, 2017. https://doi.org/10.1007/978-3-319-51281-5_25

[18] Marijan, Dusica, and Sagar Sen. "Detecting and Reducing Redundancy in Software Testing for Highly Configurable Systems." In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 96-99. IEEE, 2017. https://doi.org/10.1109/HASE.2017.31

[19] Özener, O. Örsan, and Hasan Sözer. "An effective formulation of the multi-criteria test suite minimization problem." *Journal of Systems and Software* 168 (2020): 110632. https://doi.org/10.1016/j.jss.2020.110632

[20] Alsharif, Abdullah, Gregory M. Kapfhammer, and Phil McMinn. "STICCER: Fast and effective database test suite reduction through merging of similar test cases." In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pp. 220-230. IEEE, 2020. https://doi.org/10.1109/ICST46399.2020.00031

[21] Wang, Haifeng, Bin Du, Jie He, Yong Liu, and Xiang Chen. "Ietcr: An information entropy based test case reduction strategy for mutation-based fault localization." *IEEE Access* 8 (2020): 124297-124310. https://doi.org/10.1109/ACCESS.2020.3004145

[22] Chvatal, Vasek. "A greedy heuristic for the set-covering problem." *Mathematics of operations research* 4, no. 3 (1979): 233-235. https://doi.org/10.1287/moor.4.3.233

[23] Harrold, M. Jean, Rajiv Gupta, and Mary Lou Soffa. "A methodology for controlling the size of a test suite." *ACM Transactions on Software Engineering and Methodology (TOSEM)* 2, no. 3 (1993): 270-285. https://doi.org/10.1145/152388.152391

[24] Chen, Tsong Yueh, and Man Fai Lau. "A new heuristic for test suite reduction." *Information and Software Technology* 40, no. 5-6 (1998): 347-354. https://doi.org/10.1016/S0950-5849(98)00050-0

[25] Jones, James A., and Mary Jean Harrold. "Test-suite reduction and prioritization for modified condition/decision coverage." *IEEE Transactions on software Engineering* 29, no. 3 (2003): 195-209. https://doi.org/10.1109/TSE.2003.1183927

[26] Tallam, Sriraman, and Neelam Gupta. "A concept analysis inspired greedy algorithm for test suite minimization." *ACM SIGSOFT Software Engineering Notes* 31, no. 1 (2005): 35-42. https://doi.org/10.1145/1108768.1108802

[27] Jeffrey, Dennis, and Neelam Gupta. "Test suite reduction with selective redundancy." In *21st IEEE International Conference on Software Maintenance (ICSM'05)*, pp. 549-558. IEEE, 2005. https://doi.org/10.1109/ICSM.2005.88

[28] Lin, Jun-Wei, and Chin-Yu Huang. "Analysis of test suite reduction with enhanced tie-breaking techniques." *Information and Software Technology* 51, no. 4 (2009): 679-690. https://doi.org/10.1016/j.infsof.2008.11.004

[29] Khalilian, Alireza, and Saeed Parsa. "Bi-criteria test suite reduction by cluster analysis of execution profiles." In *Advances in Software Engineering Techniques: 4th IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET 2009, Krakow, Poland, October 12-14, 2009. Revised Selected Papers 4*, pp. 243-256. Springer Berlin Heidelberg, 2012. https://doi.org/10.1007/978-3-642-28038-2_19

[30] Parsa, Saeed, Alireza Khalilian, and Yalda Fazlalizadeh. "A new algorithm to Test Suite Reduction based on cluster analysis." In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 189-193. IEEE, 2009. https://doi.org/10.1109/ICCSIT.2009.5234742

[31] Xu, Shengwei, Huaikou Miao, and Honghao Gao. "Test suite reduction using weighted set covering techniques." In *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 307-312. IEEE, 2012. https://doi.org/10.1109/SNPD.2012.87

[32] Gladston, Angelin, H. Khanna Nehemiah, Palanisamy Narayanasamy, and Arputharaj Kannan. "Test suite reduction using HGS based heuristic approach." *Computing and Informatics* 34, no. 5 (2015): 1113-1132.

[33] Cruciani, Emilio, Breno Miranda, Roberto Verdecchia, and Antonia Bertolino. "Scalable approaches for test suite reduction." In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pp. 419-429. IEEE, 2019. https://doi.org/10.1109/ICSE.2019.00055

[34] Mohapatra, Sudhir Kumar, Arnab Kumar Mishra, and Srinivas Prasad. "Intelligent local search for test case minimization." *Journal of The Institution of Engineers (India): Series B* 101 (2020): 585-595. https://doi.org/10.1007/s40031-020-00480-7

[35] Nachiyappan, S., A. Vimaladevi, and C. B. SelvaLakshmi. "An evolutionary algorithm for regression test suite reduction." In *2010 International Conference on Communication and Computational Intelligence (INCOCCI)*, pp. 503-508. IEEE, 2010.

[36] Zhang, Yi-kun, Ji-ceng Liu, Ying-an Cui, Xin-hong Hei, and Ming-hui Zhang. "An improved quantum genetic algorithm for test suite reduction." In *2011 IEEE International Conference on Computer Science and Automation Engineering*, vol. 2, pp. 149-153. IEEE, 2011. https://doi.org/10.1109/CSAE.2011.5952443

[37] You, Liang, and YanSheng Lu. "A genetic algorithm for the time-aware regression testing reduction problem." In *2012 8th International Conference on Natural Computation*, pp. 596-599. IEEE, 2012. https://doi.org/10.1109/ICNC.2012.6234754

[38] Bakar, Rohani, Kamal Z. Zamli, and Basem Al-Kazemi. "Late acceptance hill climbing based strategy for test redundancy reduction and prioritization." In *Malaysian Technical Universities Conference on Engineering and Technology (MUCET) November*, pp. 10-11. Citeseer, 2014.

[39] Zamli, Kamal Z., Mohd Hafiz Mohd Hassin, and Basem Al-Kazemi. "tReductSA–test redundancy reduction strategy based on simulated annealing." In *Intelligent Software Methodologies, Tools and Techniques: 13th International Conference, SoMeT 2014, Langkawi, Malaysia, September 22-24, 2014. Revised Selected Papers 13*, pp. 223-236. Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-17530-0_16

[40] Mohanty, Subhasish, Sudhir Kumar Mohapatra, and Sultan Feisso Meko. "Ant colony optimization (ACO-Min) algorithm for test suite minimization." In *Progress in Computing, Analytics and Networking: Proceedings of ICCAN 2019*, pp. 55-63. Springer Singapore, 2020. https://doi.org/10.1007/978-981-15-2414-1_6

[41] Coviello, Carmen, Simone Romano, Giuseppe Scanniello, and Giuliano Antoniol. "Gasser: Genetic algorithm for test suite reduction." In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1-6. 2020. https://doi.org/10.1145/3382494.3422157

[42] Xia, Chunyan, Yan Zhang, and Zhanwei Hui. "Test suite reduction via evolutionary clustering." *IEEE Access* 9 (2021): 28111-28121. https://doi.org/10.1109/ACCESS.2021.3058301

[43] Bajaj, Anu, and Om Prakash Sangwan. "Discrete and combinatorial gravitational search algorithms for test case prioritization and minimization." *International Journal of Information Technology* 13 (2021): 817-823. https://doi.org/10.1007/s41870-021-00628-8

[44] Deneke, Aliazar, Beakal Gizachew Assefa, and Sudhir Kumar Mohapatra. "Test suite minimization using particle swarm optimization." *Materials Today: Proceedings* 60 (2022): 229-233. https://doi.org/10.1016/j.matpr.2021.12.472

[45] Yoo, Shin, and Mark Harman. "Using hybrid algorithm for pareto efficient multi-objective test suite minimisation." *Journal of Systems and Software* 83, no. 4 (2010): 689-701. https://doi.org/10.1016/j.jss.2009.11.706

[46] Zamli, Kamal Z., Norasyikin Safieny, and Fakhrud Din. "Hybrid test redundancy reduction strategy based on global neighborhood algorithm and simulated annealing." In *Proceedings of the 2018 7th International Conference on Software and Computer Applications*, pp. 87-91. 2018. https://doi.org/10.1145/3185089.3185146

[47] Hooda, Itti, and R. S. Chhillar. "Test case optimization and redundancy reduction using GA and neural networks." *International Journal of Electrical and Computer Engineering* 8, no. 6 (2018): 5449. https://doi.org/10.11591/ijece.v8i6.pp5449-5456

[48] Chetouane, Nour, Franz Wotawa, Hermann Felbinger, and Mihai Nica. "On using k-means clustering for test suite reduction." In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 380-385. IEEE, 2020. https://doi.org/10.1109/ICSTW50294.2020.00068

[49] Zhong, Hao, Lu Zhang, and Hong Mei. "An experimental comparison of four test suite reduction techniques." In *Proceedings of the 28th international conference on Software engineering*, pp. 636-640. 2006. https://doi.org/10.1145/1134285.1134380

[50] Zhong, Hao, Lu Zhang, and Hong Mei. "An experimental study of four typical test suite reduction techniques." *Information and Software Technology* 50, no. 6 (2008): 534-546. https://doi.org/10.1016/j.infsof.2007.06.003

[51] Zhang, Lingming, Darko Marinov, Lu Zhang, and Sarfraz Khurshid. "An empirical study of junit test-suite reduction." In *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, pp. 170-179. IEEE, 2011. https://doi.org/10.1109/ISSRE.2011.26

[52] Noemmer, Raphael, and Roman Haas. "An evaluation of test suite minimization techniques." In *International Conference on Software Quality*, pp. 51-66. Cham: Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-35510-4_4

[53] Rahman, Mizanur, Kamal Z. Zamli, and Muhammad Arif Mohamad. "A Comparison of Four Metaheuristic Algorithms for the Problem of Test Redundancy Reduction." In *Proceedings of the 2023 12th International Conference on Software and Computer Applications*, pp. 342-348. 2023. https://doi.org/10.1145/3587828.3587879

[54] Kader, Md Abdul, Kamal Z. Zamli, and Basem Yousef Alkazemi. "An Experimental Study of a Fuzzy Adaptive Emperor Penguin Optimizer for Global Optimization Problem." *IEEE Access* 10 (2022): 116344-116374. https://doi.org/10.1109/ACCESS.2022.3213805