



Applied Artificial Intelligence

An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: www.tandfonline.com/journals/uaai20

Unsupervised Machine Learning Approaches for Test Suite Reduction

Anila Sebastian, Hira Naseem & Cagatay Catal

To cite this article: Anila Sebastian, Hira Naseem & Cagatay Catal (2024) Unsupervised Machine Learning Approaches for Test Suite Reduction, Applied Artificial Intelligence, 38:1, 2322336, DOI: [10.1080/08839514.2024.2322336](https://doi.org/10.1080/08839514.2024.2322336)

To link to this article: <https://doi.org/10.1080/08839514.2024.2322336>



© 2024 The Author(s). Published with license by Taylor & Francis Group, LLC.



Published online: 04 Mar 2024.



Submit your article to this journal [↗](#)



Article views: 5553



View related articles [↗](#)






View Crossmark data [↗](#)



Citing articles: 1 View citing articles [↗](#)

Unsupervised Machine Learning Approaches for Test Suite Reduction

Anila Sebastian , Hira Naseem , and Cagatay Catal 

Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar

ABSTRACT

Ensuring quality and reliability mandates thorough software testing at every stage of the development cycle. As software systems grow in size, complexity, and functionality, the parallel expansion of the test suite leads to an inefficient utilization of computational power and time, presenting challenges to optimization. Therefore, the Test Suite Reduction (TSR) process is of great importance, contributing to the reduction of time and costs in executing test suites for complex software by minimizing the number of test cases to be executed. Over the past decade, machine learning-based solutions have emerged, demonstrating remarkable effectiveness and efficiency. Recent studies have delved into the application of Machine Learning (ML) in the software testing domain, where the high cost and time consumption associated with data annotation have prompted the use of unsupervised algorithms. In this research, we conducted a Systematic Mapping Study (SMS), examining the types of unsupervised algorithms implemented in developed models and thoroughly exploring the evaluation metrics employed. This study highlighted the prevalence of the K-Means clustering algorithm and the coverage metric for validation in various studies. Additionally, we identified a gap in the literature regarding scalability considerations. Our findings underscore the effective use of unsupervised learning approaches in test suite reduction.

ARTICLE HISTORY



Received 7 November 2022

Revised 29 November 2023

Accepted 30 January 2024

Introduction

With the massive increase in software dependency, there is a growing demand for the reliability of software to prevent catastrophes and financial losses resulting from software failures (Iannino and John 1990). To ensure software quality, software testing plays a crucial role and is among the most extensively used practices for assessing and improving software quality (Orso and Rothermel 2014). Various domains of software testing have been explored to ensure reliability, making it one of the most focused domains in software engineering conferences (Lewis 2004). One prominent problem in this domain

CONTACT Cagatay Catal  ccatal@qu.edu.qa  Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha 2713, Qatar

© 2024 The Author(s). Published with license by Taylor & Francis Group, LLC.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

can be described as follows: *“Provided a system S, with an associated test suite T, how it could be verified whether S passes on T, or -if not- then identify the failing test cases, in such a way that resources utilized to execute T on S should be managed efficiently.”* In this description, the focal term is “efficiently”: Therefore, the simplest solution would be to execute S on T. The research in this domain aims that S needs to be repetitively tested on T (known as regression testing) and the retest-all approach might be expensive in terms of available resources such as computational power.

As a part of this process, test suites are used to check the correctness of the system under test (SUT). However, as the size, complexity, or functionality of the software increases, the size of the test suite also grows. The resources required to execute this large number of test cases are limited, making this task time-consuming and computationally expensive. Therefore, it is desirable to identify a subset of these test cases for efficient resource management (Martins et al. 2021). For example, if the case of regression testing is considered, where testing is done to ensure that the system has not been affected by a recent modification, it is better to find the test cases related to the recent changes (Al-Sabbagh et al. 2020). This process of identifying a smaller subset of the test suite is known as test suite minimization (Jehan and Wotawa 2023), test suite optimization (Gupta, Sharma, and Pachariya 2022), and sometimes, test suite reduction (Habib, Khan, and Felix 2023). In this study, this process is referred to as Test Suite Reduction (TSR).

The TSR problem can be stated as follows (Harrold, Gupta, and Lou Soffa 1993): *“Test suite T, and a set of test-case requirements $r_1; r_2 \dots r_n$ that must be satisfied to provide the desired test coverage of the program, and subsets of T, $T_1; T_2 \dots T_n$, one associated with each of the r_i s such that any one of the test cases t_j belonging to T_i can be used to test r_i ”*. While selecting a subset of a test suite, the main challenge is to identify and select those test cases that can identify the bugs and faults in the system under test (SUT). Otherwise, the reduced test suite will fail to achieve its primary goal of fault identification. Many studies have employed different techniques to achieve this desired goal of TSR. There have been various approaches in practice such as adaptive random testing, search-based methods, greedy algorithms, and fuzzing.

One of the major drawbacks of these practices is that the outputs of the test cases need to be observed manually. This approach could be time-consuming, costly, and unreliable, and the output produced in these techniques might be very large (Shahamiri, Mohd Nasir Wan Kadir, and Zaiton Mohd-Hashim 2009). The solution to this problem is to provide a reliable test suite oracle, which can compare the expected and received output; however, a reliable test oracle is very unlikely to be provided (Miller, Fredriksen, and Bryan 1990). Another drawback of such methods is that they can identify and detect very limited faults. Detecting security vulnerabilities through memory leaks or buffer overflows has very limited advantages with these techniques compared

to their allocated resources. According to a research article, for a software project, around 50% of the total budget is allocated for testing (Harrold 2000). Allocating these resources for exhaustive testing (i.e., during Continuous Integration (CI) and regression testing) is very costly. Moreover, despite being expensive, this practice is unavoidable to fulfill certain standards and quality assurance criteria. For instance, in safety-critical software systems, comprehensive testing is mandatory, and this testing must adhere to the guidelines provided by different standards (Holcombe, Ipate, and Grondoudis 1995).

The resources required to check the bugs or faults in the software are directly proportional to the complexity of the software. Consequently, the complexity of the software increases the computational cost of the testing, and each testing session delays the work of developers (Mottaghi and Reza Keyvanpour 2018). The identified problem requires an effective and efficient solution. A solution must not only pinpoint all the bugs in the software but also reduce testing time. Addressing this problem can significantly contribute to the efficient testing of the software development industry; thus, it cannot be neglected (Al-Sabbagh et al. 2020). The concept of TSR was first proposed by Harrold, Gupta, and Lou Soffa (1993) (Harrold, Gupta, and Lou Soffa 1993). In their study, they proposed a method to select a representative subset of test cases from a test suite that could provide the same coverage as the whole test suite. To achieve this, redundant and useless test cases from the test suite were identified and eliminated. Chen and Lau also conducted a study on TSR (Chen and Lau 1998). According to that study, some subsets of a test suite may have the capability to satisfy the same testing objectives, and these were referred to as representative sets of the test suite.

If the SUT is exceptionally large or deeply interacts with its environment, testing becomes a highly time-consuming process (Martins et al. 2021). Executing and validating large test suites for such SUTs within predetermined time limits poses a significant challenging. Hence, identifying representative sets of the test suite that can detect faults within the given timeframe is desirable. Another approach to handle this problem is test case prioritization (Catal and Mishra 2013), which executes those test cases that are more likely to fail (Rothermel et al. 1999). However, this study focuses on test suite reduction, and therefore, the prioritization of test cases is not within the scope of this paper. TSR can be of two types: adequate and inadequate (Shi et al. 2014). The first category consists of approaches that reduce the test suites in a way that completely satisfies the test requirements of the original test suites. The latter category includes approaches that produce a reduced test suite; however, this test suite only partially preserves the test requirements of the original test suite.

Machine learning is linked to the exploration, identification, and learning of patterns from given data (Azuaje, Witten, and Frank 2006). These patterns can be utilized to extract information about the domain and aid in decision-

making. Machine learning is identified as one of the most desirable approaches for some software quality assurance activities according to a study (Catal 2011), which demonstrates that machine learning-based solutions can outperform classic approaches in this domain due to their higher probability of fault detection. Machine learning approaches are not constrained by fundamental theory, assumptions, and model representations. Their algorithms vary in origin, ranging from traditional statistics to heuristic learning algorithms and neural networks. However, machine learning approaches can mainly be categorized into two broad types: supervised machine learning and unsupervised machine learning. Supervised learning approaches require labeled data, where classified data is fed into the algorithm. In contrast, unsupervised approaches do not require annotation for the model to learn and extract patterns from the dataset. The learning model forms clusters of data based on certain features (K. Khan et al. 2014). This approach has a clear advantage over the supervised approach, as annotating a dataset requires the same resources and efforts as training a model. Hence, overhead can be avoided by utilizing this technique. A recent tertiary study also investigated the use of Artificial Intelligence in software testing, and it did not solely focus on machine learning algorithms (Amalfitano et al. 2023).

Our Systematic Mapping Study (Ahmad et al. 2023; Breit et al. 2023; Petersen et al. 2008) thus focuses on the unsupervised approaches chosen for TSR. To systematically map the literature, we established inclusion and exclusion criteria, employed a search strategy, and followed a systematic process to categorize and map the selected studies. This approach allowed us to capture the breadth of research in this field. For this purpose, we reviewed recent studies in this area using the research questions outlined in Table 1.

The following sections are organized as follows: Section 2 includes the background and the related work in the given domain, Section 3 states the methodology followed to construct this article, Section 4 contains the results obtained from reviewed research articles, Section 5 presents the discussion, and Section 6 presents the conclusion and future work of this study.

Background and Related Work

Unsupervised Learning for Test Suit Reduction

Machine Learning (ML) is a sub-branch of Artificial Intelligence (AI) that studies algorithms and methods for automating solutions to complex problems, which are challenging to program using conventional programming methods (Rebala, Ravi, and Churiwala 2019). There are four types of learning approaches in ML: supervised, unsupervised, semi-supervised, and reinforcement learning. In supervised ML, a model is generated first by processing a labeled dataset, which is later used to label any new input data points.

Table 1. Research questions and main motivations.

No.	Research Question	Motivation
RQ-1	What technique/algorithm has been used?	The formation and quality of clusters can affect the coverage. Selecting no less than one test case from every cluster inside the test suite can provide the same coverage through a smaller number of test cases (Golagha et al. 2019). This question helps to understand the methods adopted by the research community.
RQ-2	What evaluation metric was used in the paper to measure the effectiveness of the approach?	The reliability of software is of keen importance (Iannino and John 1990) to ensure that regression testing plays a key role. If a test suite fails to find faults in the software, then it can be referred to as inadequate (Coviello et al. 2020). The efficacy of a test suite can be measured using different evaluation metrics. Similarly, repeating the experiment and calculating the average efficacy can result in the efficacy of the model developed by the researcher. To compare the performance of the model with a benchmark, a standard or commonly used metric is needed. This directed us to this research question to identify metrics, which have been widely used and can be chosen as a benchmark in studies.
RQ-3	What kind of artifacts were shared with the research community (e.g., dataset/source code/model)?	The research community builds their work on other studies to improve the state-of-the-art instead of starting from scratch in each research. Hence, the artifacts developed during research are mostly shared with the community to increase the acceptance of their work. This also aids in the reusability of developed artifacts. Thus, this research question was deduced to give information about which studies have made their models and data publicly available. This can aid in novel research using pre-developed models.
RQ-4	How scalable is the proposed approach?	Another aspect of machine learning-based models is that they need a training dataset and time for training. It must be studied if the established model is reasonable and scalable for real-world applications. This statement shaped our next research question for the SLR to access the scalability of different approaches. This helps researchers understand the need for more scalable systems, which can contribute to this field.
RQ-5	What domains have been explored?	There are many domains of software applications such as web-based applications, plug-ins for available applications, APIs, databases, embedded systems, and many more. In this study, the software application domains in which unsupervised TSR is effective have also been explored. The most frequent domain used in this approach helps to explore new avenues, which have not been explored yet.
RQ-6	What were the objectives of the research?	There are several factors which motivate TSR, such as cost reduction, time, or resource constraints. The driving factors of the research in the given article.
RQ-7	What were the challenges presented in the field and proposed solutions?	In the given domain, the knowledge gaps were identified by the researchers. These gaps motivate the need of solution and then certain strategies or models were designed to fulfill the gap.
RQ-8	Which feature selection or feature reduction techniques were utilized?	In ML, the main contributor is the dataset. By utilizing feature selection or feature reduction techniques effectively, the efficacy and efficiency can be enhanced for the learning models.

Support Vector Machines (SVM), Decision Trees, and Linear Regression are a few algorithms that fall under this category. This approach requires a well-labeled training dataset. Annotating data in large amounts is time-consuming, costly, and sometimes not possible (Sun, Chenchun, and Suominen 2021). To handle this issue, an unsupervised machine learning approach is applied.

Unsupervised learning does not require a labeled dataset. The algorithms in this category aim to discover existing natural patterns within the training dataset. Some common examples include clustering algorithms and feature selection techniques such as Principal Component Analysis (PCA). Clustering algorithms like K-Nearest Neighbor (KNN) aim to group training examples based on the similarity of their features. In Principal Component Analysis (PCA), the algorithm attempts to reduce the number of features by identifying features that can discriminate between training examples. This is used in the data pre-processing (or data preparation) phase. PCA also reduces the dimensions of training data, making the vector representation denser. Many unsupervised and clustering approaches are efficiently used by researchers for TSR. In the case of using clustering, a program can be tested using any clustered test case instead of the entire test cases in the test suite (Kiran et al. 2019). Clustering groups similar test cases into the same clusters and less-similar test cases into different clusters. These clusters are determined with the help of dissimilarity metrics like Euclidean distance or Manhattan distance. Moreover, this technique allows using various test information along with the inputs and outputs of programs to determine such clusters based on the tester's available resources and objectives [P1].

Related Work

This SMS has chosen to study the relevant literature in the field of unsupervised approaches for test suite reduction. Few existing studies have focused on topics related to this issue. Prior research has touched upon various aspects of test suite optimization, as exemplified by Kiran et al. (2019) who focused on the latest test suite optimization tools, trends, and techniques. In their article, 58 studies published from 2016 to 2019 were reviewed and grouped into five categories. They also summarized information regarding 32 prominent tools used by such studies. The authors highlighted the underrepresentation of clustering-based techniques in the existing body of work

In a study by S. U. R. Khan et al. (2016), a thorough evaluation of Test Suite Reduction (TSR) frameworks and tools was conducted. Employing a thematic arrangement, the authors categorized the existing literature, providing insights into the evolving landscape of TSR methodologies. Furthermore, Mottaghi and Reza Keyvanpour (2018) delved into studies related to TSR utilizing data mining approaches. Their categorization included techniques such as classification, cluster analysis, and mining frequent itemsets. Notably, the cluster

analysis techniques were further stratified into coverage-based, density-based, similarity-based, and a combination of coverage-based and distribution-based methods, offering a nuanced perspective on the diverse approaches within this domain.

In an assessment conducted by Rehman et al. (2018), an in-depth evaluation of Test Suite Reduction (TSR) approaches and associated experiments was undertaken. Their study not only scrutinized the quality of existing TSR methodologies but also offered valuable guidelines. Recognizing a lack of standardized practices in planning, conducting, and reporting experiments in this field, the authors sought to provide a structured framework. This investigation, encompassing 113 articles published between 1993 and 2016, contributes significantly to the understanding and enhancement of experimental practices within the TSR domain. Alkawaz and Silvarajoo (2019) conducted an empirical study on Test Case Prioritization and Optimization Techniques in software regression testing. Their primary focus centered on test case prioritization, with only one of the five research questions in their study dedicated to test suite optimization. While their emphasis leaned toward prioritization, their work contributes insights into the broader landscape of techniques employed in software regression testing. Another similar review of test case prioritization and optimization techniques was undertaken by Saraswat, Singhal, and Bansal (2019). Their study primarily focused on the analysis of evaluation metrics employed in these studies, with a specific emphasis on investigating the use of the genetic approach within this context. In a comprehensive analysis, Anwar and Ahsan (2014) conducted a detailed study on regression test suite optimization. Their analysis delved into a range of aspects, including tools, techniques, mathematical models, objective functions, and identified research gaps in this field. Furthermore, they offered valuable insights on obtaining sample programs and test suites for conducting experiments within this domain. Yoo and Harman (2012) conducted a review of heuristics for test suite minimization. Their exploration not only encompassed an analysis of various heuristics but also investigated the impact of these techniques on the fault-detection capabilities of test suites, providing valuable insights into the efficiency and effectiveness of test suite minimization strategies. Ali et al. (2019) performed a Systematic Literature Review (SLR) considering industry-relevant regression testing research. They mapped 38 papers explaining 26 regression testing techniques in industrial settings and proposed three taxonomies to compare the applicability of regression testing techniques.

Through a review of related works in this domain, we observed that that no prior research has investigated the specific topic currently explored in this SMS. Notably, existing literature emphasizes the significance of clustering as a pivotal technique in Test Suite Reduction (TSR). Consequently, conducting a targeted review, with a primary focus on the application of unsupervised/

clustering techniques in TSR, is deemed valuable. This approach aims to identify research gaps, propose potential solutions, and offer an exploration of the prevailing algorithms and metrics commonly employed in this field, providing valuable insights for researchers in this area.

Research Methodology

Kitchenham and Charters proposed a series of steps to be followed to conduct an effective review study. Their proposed methodology shown in [Table 2](#) has been applied as a guideline to structure our study (Kitchenham and Charters 2007). Their strategy comprises three main phases and each phase in turn consists of several steps.

[Figure 1](#) depicts the various steps that were followed by the authors in this study.

Primary Study Selection

Various online databases including IEEE Xplore, Science Direct, ACM Digital Library, Scopus, Springer Link, and Wiley Online Library were utilized in this study. A few research papers were also selected manually by extracting from the reference section of research papers (a.k.a., snowballing). Studies were limited to those published during the period 2013–2023. Different combinations of keywords were investigated. The queries used were as follows:

IEEE Xplore, Science Direct, Scopus, Springer Link, and Wiley Online Library: (“test suit reduction” OR “test suite reduction” OR “test suit minimization” OR “test suite minimization” OR “test suite optimization” OR “test suite optimization”) AND (“clustering” OR “unsupervised”)

ACM Digital Library: [[All: “test suit reduction”] OR [All: “test suite reduction”] OR [All: “test suit minimization”] OR [All: “test suite minimization”] OR [All: “test suit optimization”] OR [All: “test suite optimization”]] AND [[All: “clustering”] OR [All: “unsupervised”]]

Table 2. Kitchenham and charters’ methodology.

Phase	Proposed Steps
Planning the Review	Identification of the novel topic for review
	Commissioning a review (Optional)
	Specifying the research question(s)
	Developing a review protocol
	Evaluating the review protocol (Optional)
Conducting the Review	Identification of research
	Selecting the research articles (primary studies)
	Describing criteria for quality assessment
	Data extraction and monitoring
	Data synthesis
Reporting the Review	Specifying dissemination mechanisms
	Formatting the main report
	Evaluating the report (Optional)

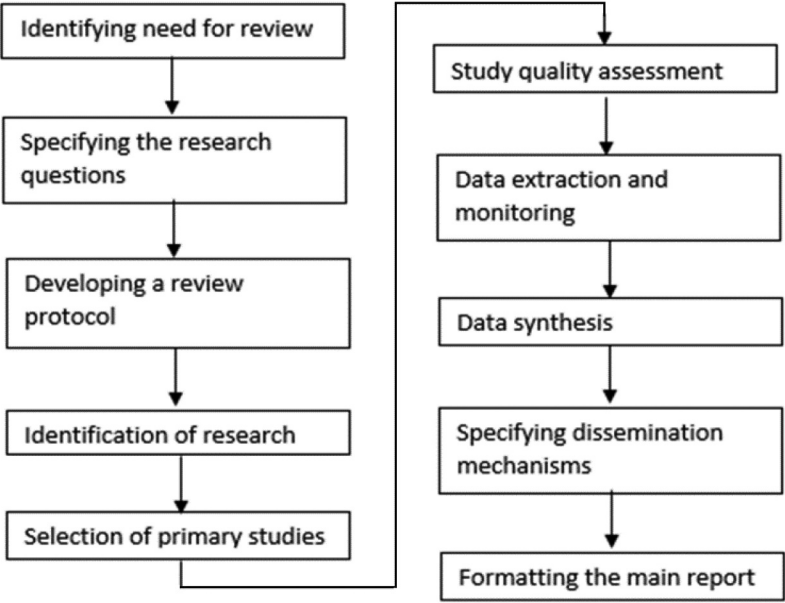


Figure 1. Methodology of this study.

Table 3. Exclusion criteria.

1	Research papers that are not related to testing optimization and do not use an unsupervised learning approach
2	Research papers that are written in languages other than English
3	Duplicate research papers or those that were already retrieved from another database
4	Review and survey papers
5	Papers that have only abstract

Table 4. Quality assessment criteria.

Q-1:	Are the aims of the study clearly stated?
Q-2:	Are the scope and experimental design of the study defined clearly?
Q-3:	Are the variables in the study likely to be valid and reliable?
Q-4:	Is the research process documented adequately?
Q-5:	Are all the study questions answered?
Q-6:	Are the negative findings presented?
Q-7:	Are the main findings regarding creditability, validity, and reliability stated?
Q-8:	Do the conclusions relate to the aim or purpose of the study?

The exclusion criteria provided in Table 3 were applied to the query results retrieved from various sources. After this step, a quality evaluation based on questions given in Table 4 was performed. None of the papers was removed after the quality assessment because all the papers score more than 4 points, which was the threshold value. The research papers were assigned a score of 1 for “Yes,” .5 for “Partial,” and 0 for “No.” The results of this quality assessment can be seen in Figure 2. The paper selection process according to the number of articles in each stage is shown in Table 5.

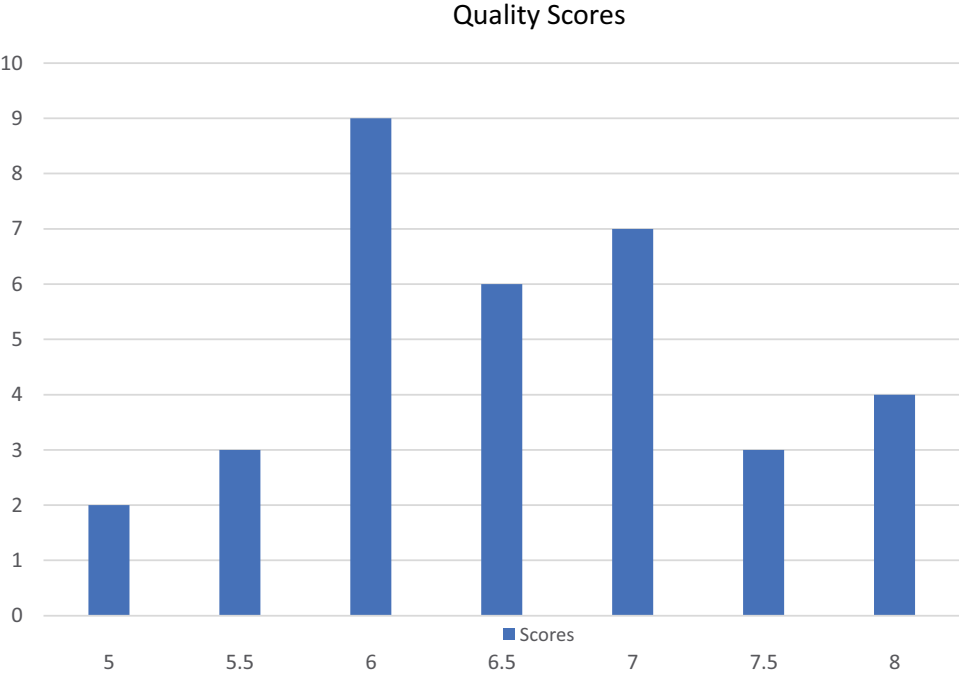


Figure 2. Quality scores of research papers.

Table 5. Paper selection process.

Source	After query search	After applying the selection criteria	After quality assessment
IEEE Xplore	15	7	7
Science Direct	75	3	3
ACM Digital Library	53	7	7
SCOPUS	34	10	10
Springer Link	69	4	4
Wiley Online Library	8	1	1
Manual	2	2	2
Total	254	34	34

The distribution of research papers according to their sources can be found in [Figure 3](#). As shown in this figure, the most used database is SCOPUS. Also, IEEE and ACM digital library provided a similar number of relevant papers in this research.

Data Extraction

To find the answers to the research questions, relevant data had to be extracted from the 34 research papers identified in the previous steps. Data responding to each primary study in this study was collected in the form of a spreadsheet. In the spreadsheet, each research paper was assigned a specific row and each answer was specified in a specific column. [Table 6](#) shows the data extraction form that was used for this purpose.

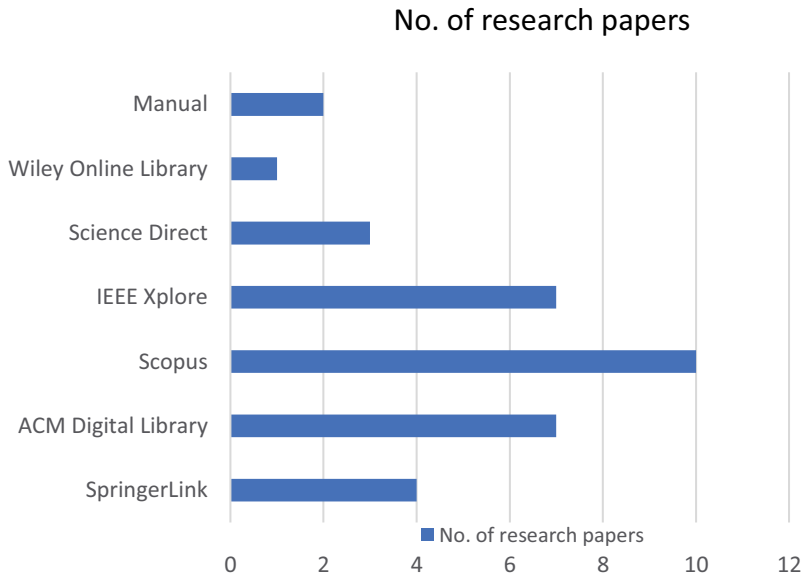


Figure 3. Distribution of research papers according to their sources.

Table 6. Data extraction form.

No.	Criteria
1	Title
2	Database
3	Keywords
4	Year of publication
5	DOI
6	Algorithm used
7	Metrics used
8	Problem description
9	Method
10	Future research directions
11	Availability of the artifacts for the research community
12	The scale of the approaches

Data Synthesis and Reporting

The data collected during the data extraction phase was aggregated during the data synthesis stage. For the first four research questions, the extracted data was divided into several categories. In the case of RQ-1, the categories included K-means clustering, agglomerative clustering, hierarchical clustering, fuzzy c-means clustering, DBSCAN, *Hierarchical Agglomerative Clustering* algorithm, *Clustering with SLOPE* algorithm, and Formal Concept Analysis (FCA), K-medoids algorithm, Expectation-Maximization (EM) and others. For RQ-2, various evaluation metrics were considered, such as coverage, mutation score, fault detection effectiveness, fault detection loss, precision, recall, F-measure, accuracy, time, and others. For RQ-5, the domain was divided into different categories, such as web-based and distributed.



Figure 4. Number of research papers published per year.

Results

This section presents the results of the research questions obtained after conducting the previous steps. A total of 254 research papers were retrieved from different databases and 34 papers were selected for this study. [Appendix A](#) shows the details of these 34 papers. [Figure 4](#) illustrates the count of papers published per year. It was easily noticeable from the results that there are more papers published in recent years, indicating that this is still an active research field. Out of the total 34 papers, 20 papers were published after 2017. As the search period concluded in early 2023, we do not see the whole papers published in 2023.

RQ-1: What Technique/Algorithm Has Been Used?

Several unsupervised algorithms/techniques used in the 34 papers were categorized into the following categories:

- *K-Means clustering*- This clustering algorithm was initially introduced by James MacQueen (on and 1967, [n.d.](#)). Unlabeled data is first partitioned into k (a previously determined value) clusters. Initially, there are k centroids and the algorithm iteratively assigns elements to that cluster with the closest mean based on a distance metric (e.g., Euclidean/ Manhattan distance). After the allocation of elements to clusters, new cluster centroids are calculated. Finally, the algorithm stops when cluster partitions become stable. Our study found that K-means clustering is the most widely used unsupervised approach for TSR. [P16] used Lloyd’s algorithm for K-Means clustering and [P4] used K-Means++ clustering.

- *Hierarchical Clustering*- This unsupervised ML approach is also known as Hierarchical Cluster Analysis (HCA) (“Oded Maimon and Lior Rokach. 2010. Data Mining and... - Google Scholar,” n.d.). Data objects are organized into a tree structure by this distance-based technique. Here, if m is the number of distinct data objects, the number of clusters will be in the range of 1 to m .
- *Agglomerative Clustering*- This type of clustering can be considered a “bottom-up” approach to Hierarchical Cluster Analysis (HCA). Initially, each data object is allocated to its cluster. Then, in every step, a negligibly dissimilar pair of clusters is selected and combined to become a cluster, which replaces the chosen pair. For a predefined value “ k ,” this process is repeated until there are “ k ” clusters (Dickinson, Leon, and Podgurski 2001).
- *The fuzzy c-means clustering*- This algorithm is prominent and extensively used for clustering. In the first step, it chooses the number of clusters. Then, all data points are assigned coefficients randomly for being part of the cluster. Until the convergence of the algorithm, centroids are found for each cluster, and coefficients are found for all data points. The convergence of the algorithm is determined by checking whether there is a change in coefficients between two iterations that do not exceed a given sensitivity threshold [P28].
- *DBSCAN (Density-Based Spatial Clustering of Applications with Noise)* – DBSCAN, which is one of the first density-based clustering algorithms, was intended to cluster data of random shapes in spatial and non-spatial high dimensional databases (K. Khan et al. 2014). The main idea for this algorithm is that for each data object in a cluster, the locality of a specified radius should contain the least number of objects. If this is not true for an object p , a new cluster is created, and p is assigned as a core point.
- *Hierarchical Agglomerative Clustering (HAC) algorithm*- This algorithm builds clusters by merging based on proximity and existing clusters. An initial set of clusters is selected first. Then, the nearest pair of clusters are found and merged. This step gets repeated until a “stopping criteria” gets satisfied. This criterion can be reaching a number of clusters, attaining a pre-defined number of elements in each cluster. A combination of such criteria can also be used (Yager 2000).
- *CLOPE (Clustering with SLOPE) algorithm*- This algorithm is fast and competent for clustering bulky and high dimensional data (Khalilian and Parsa 2009). It controls the level of intra-cluster similarity by setting a value called “Repulsion.” The number of clusters can be varied by making changes to this value.
- *Formal Concept Analysis (FCA)*- This method primarily derives implied associations between objects defined through a set of attributes and these attributes. The data is arranged into parts that are proper abstractions of

notions of human thought, which allows expressive understandable interpretation(“B. Ganter and R. Wille. Formal Concept Analysis -. . . - Google Scholar,” n.d.).

- *K-Medoids Algorithm*- If N is a data set and k – the number of clusters, then k data elements from N are arbitrarily selected as centers of initial clusters. After this, the distance amid the data elements of the non-center points in N and the centers of the clusters are found. According to these values, the outstanding data elements are allocated to the adjacent clusters. The center point is replaced by re-using the non-central points in the cluster[P19].
- *Expectation-Maximization (EM)*- In this approach, one or more probability distributions are used to compute probabilities of cluster memberships (Dempster, Laird, and Rubin 1977). Given the final clusters, the goal is to maximize the total probability of the data points. It allows data points to be part of more than 1 cluster (unlike most other clustering algorithms). A maximum number of interactions or a measure of the quality of the composed clusters can be used as a stop criterion. This clustering algorithm can be used as a supervised and semi-supervised approach too but the 2 studies in this study have used it as an unsupervised approach.
- *Others*- Some unsupervised approaches which were not used in more than 1 paper and algorithms for which the authors did not specify any names were included in this section. This includes many algorithms such as Affinity Propagation, Principal Component Analysis (PCA), K-nearest neighbor, cluster searching algorithm, subtractive clustering, Maximal frequency item set clustering, and Most Maximal Frequent Trace Clustering. One research article utilized Dissimilarity-based Sparse Subset Selection in their study. These mentioned algorithms are included in this category.

Table 7 provides references to the primary studies based on the algorithms they used. Most of the papers have used the K-means clustering approach (i.e., 12). This was followed by Hierarchical Clustering Algorithm (i.e., 3).

Table 7. Algorithms used in articles.

Algorithm	References to Papers that used it
K-Means clustering	[P1], [P2], [P4], [P11], [P20], [P26], [P27], [P32], [P34], [P25], [P23], [P16]
Hierarchical Clustering	[P17], [P30], [P26]
Agglomerative Clustering	[P6]
Fuzzy c-means clustering	[P28]
DBSCAN	[P7], [P35]
Hierarchical Agglomerative Clustering	[P15], [P21], [P22]
CLOPE	[P10]
Formal Concept Analysis	[P9]
K-Medoids Algorithm	[P19], [P27]
Expectation Maximization	[P1], [P33]
Others	[P1], [P4], [P5], [P31], [P14], [P29], [P24], [P18], [P13], [P8], [P3], [P25]

RQ-2: What Evaluation Metric Was Used in the Paper to Measure the Effectiveness of the Approach?

Various metrics used by the different papers in this study were categorized into the following categories:

- *Coverage*- This metric includes several coverage metrics like branch coverage, statement coverage, and code coverage. Branch coverage refers to “the percentage of branches that have been evaluated to be both true and false by the test suite.” Statement coverage is “the proportion of the number of statements executed by the test suite to the total number of existing statements in the SUT”(Felbinger, Wotawa, International, and 2017, [n.d.](#)). Code coverage gives a measure of the size of code executed by the test suit.
- *Mutation Score*-Faulty program versions are created by introducing mutations to the program in mutation testing. A mutant gets killed if any test case in the test suite can produce an unexpected result when executing the mutant. Mutation Score refers to the proportion of killed mutants to the total number of generated mutants (Felbinger, Wotawa, International, and 2016, [n.d.](#)).
- *Fault detection effectiveness*- This metric measures the effectiveness.
- *The fault detection loss*-This metric is the ratio between the difference in faults detected by the initial test suite and the newly generated test suite to the number of faults detected by the initial test suite.
- *Precision*-Precision can be measured as the ratio between True Positives and the sum of True Positives and False Positives.
- *Recall*-Recall can be defined as the ratio between True Positives and the sum of True positives and False negatives.
- *F-measure* -It is the harmonic mean of precision and recall.
- *Test suite size reduction*: This indicates the reduction capability of the model.
- *Accuracy*-It is the well-known accuracy metric used in machine learning studies.
- *Time*-It measures the time required to execute the reduced test suite.
- *Others*-All other metrics that have not been used in more than one paper were included in this category (e.g., entropy, mean squared error).

Table 8 summarizes the distribution of the use of different metrics in the selected papers. Also, it provides references to the primary studies based on the metrics they used. It was noted that the largest number of papers used the “Coverage” metric. Some other metrics have not been mapped into one of these metric categories, as such, they were listed under the *others* category.

Table 8. Evaluation metrics used in the research articles.

No.	Metric	No. of Research Papers	References
1	Coverage	10	[P1], [P2], [P19], [P20], [P34], [P27], [P13], [P16], [P23], [P24]
2	Mutation score	3	[P2], [P5], [P17]
3	Fault detection effectiveness	3	[P32], [P3]
4	Fault detection loss	3	[P4], [P22], [P8]
5	Precision	4	[P6], [P9], [P31], [P33]
6	Recall	3	[P6], [P9], [P33]
7	F-measure	2	[P6], [P33]
8	Test suite size reduction	10	[P32], [P4], [P33], [P21], [P10], [P14], [P8], [P13], [P22], [P26]
9	Accuracy	1	[P11]
10	Time	6	[P4], [P35], [P13], [P16], [P18], [P24]
11	Others	20	[P6], [P9], [P33], [P35], [P7], [P11], [P20], [P28], [P15], [P17], [P21], [P10], [P14], [P3], [P13], [P24], [P25], [P26], [P29], [P30]

RQ-3: What Kind of Artifacts Were Shared with the Research Community (E.G., Dataset/Source Code/Model)?

The results of this research question were divided into the following categories:

- *Yes*-This includes those studies, which made available different artifacts that are necessary to replicate their work. This includes artifacts such as source code, frameworks, datasets, and scripts.
- *No*-This category consists of those studies, which did not make the artifacts related to their work publicly available.
- *Pseudocode/Algorithms*-This category includes those studies which made pseudocode/their novel algorithms available through their publication.

Figure 5 summarizes the answer to this RQ-3. Only three research papers ([P4], [P7], [P14]) have made their entire framework/code available along with the data used. 14 research papers ([P2], [P6], [P3], [P11], [P17], [P22], [P24], [P25], [P16], [P18], [P33], [P30], [P34], and [P35]) have provided their pseudocode/novel algorithms in their paper. This figure indicates that most researchers do not share research-related artifacts with the research community in this field.

RQ-4: How Scalable is the Proposed Approach?

Figure 6 summarizes the answer to RQ-4. Three research papers ([P6], [P7], and [P15], and) have mentioned that their approach is scalable and one of them ([P9]) mentioned that their approach is not scalable. Two research papers ([P2], [P4]) have mentioned that they need to

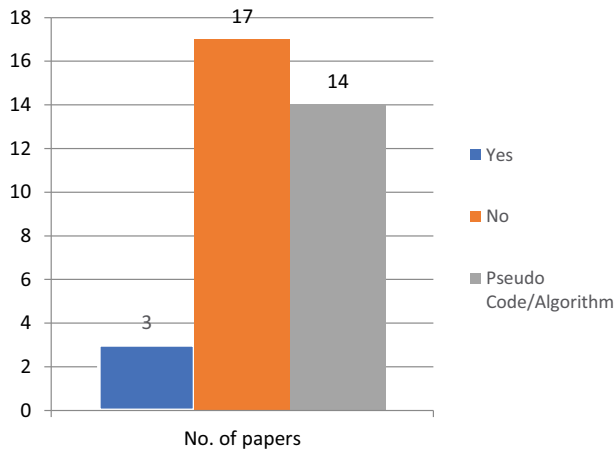


Figure 5. Availability of artifacts in research articles.

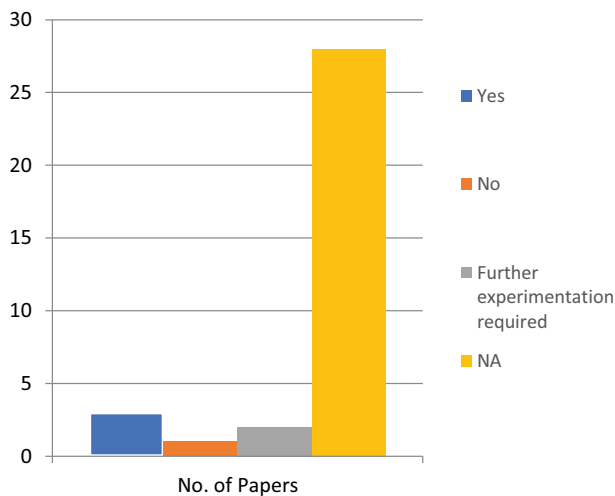


Figure 6. Feasibility to scale.

perform more experiments to know more about the scalability of their approach.

Based on this figure, scalability has been overlooked in most of the research.

RQ-5: What Domains Have Been Explored?

It was observed that TSR techniques are not specific only to academia. Many studies have tested them on real-world applications as well. From web-based applications to system testing, from distributed systems to embedded systems different software domains are utilizing these solutions.

These models are aiding in test suite reduction at each level of testing (e.g., unit testing, integration testing, mutation, end-to-end testing, and system testing). To be specific (Sampath and Bryce 2012, n.d.), and [P26] reduced the test suite for web-based applications using unsupervised learning models. The detailed domain of SUTs mentioned in research articles is given in Table 9.

Other studies mention profiling and performance testing in operating systems [P6], plugins designed for Eclipse 4.2 was tested in [P15], and functional testing of components of SUT was done in [P1]. Some research articles (Hsueh, Pin Cheng, and Cheng Pan 2011) conducted end-to-end testing on GUI events and replays using this TSR approach. The authors in the studies [P29] and [P23] conducted their experiments in database applications. Whereas the authors in the study [P30] focused on Side-Channel Vulnerability Detection. Test optimization problems from the telecommunication and maritime domains were chosen by the authors in the study [P16]. In short, this technique can be incorporated at any level or any domain of software testing.

RQ-6: What Were the Objectives of the Research?

The main objective mentioned in the articles was TSR for effective utilization of time and resources. These ML based solutions were used with test suites of different testing techniques such as Regression testing, Mutation testing, Combinatorial testing, Black box testing and White box testing. Some articles do not mention the type of testing and kept it

Table 9. Domain of SUTs in research articles.

No.	Domain	References
1	Web-based	[P26]
2	Distributed systems	[P11], [P6], [P5]
3	AI-based systems	[P7]
4	Industrial software (using Agile Development)	[P31], [P27]
5	Open-source dataset for academia	[P14], [P17]
6	Database-based	[P29], [P23]

Table 10. Type of testing explored in each article.

No.	Type of Testing	References
1	Regression testing	[P1], [P4], [P5], [P8], [P10], [P12], [P13], [P14], [P15], [P17], [P20], [P21], [P24], [P26], [P29], [P31], [P32]
2	Mutation testing	[P2], [P17]
3	Combinatorial testing	[P18]
4	Black box testing	[P34]
5	White box testing	[P34]
6	Unit testing	[P33]
7	Overall software testing	[P7], [P9], [P11], [P12], [P16], [P19], [P25], [P27], [P28], [P30]

Table 11. Objectives mentioned by the authors in their article.

No.	Objectives (TSR)	References
1	Consider functional aspect of software for TSR.	[P1]
2	Find appropriate quantity of clusters to reduce the test suite without deviating from coverage or mutation score.	[P2]
3	Find minimal test suite T' from T which satisfies all test requirement ri covered by original suite T.	[P3]
4	Regression testing for massive systems.	[P4]
5	Regression testing for self-organizing software.	[P5]
6	Extensive software testing in automotive industry.	[P6]
7	Efficient Deep Neural Network (DNN) testing for DNN models.	[P7]
8	Maximum statement coverage in minimum execution time.	[P8]
9	Fault localization during software testing	[P9]
10	Comparison of adequate and inadequate TSR approaches with respect to the size of reduced test suites and their fault-detection capability	[P10]
11	Test suite reduction to enable recovery of the faults in the software in limited time	[P11]
12	Eliminating test cases which do not aid in fault detection during regression testing	[P12]
13	Test suite optimization for regression testing in a cost-effective manner.	[P13]
14	Optimize the resource consumption in regression testing.	[P14]
15	Develop a Plugin for Eclipse to eliminate redundant test cases in regression testing.	[P15]
16	Reduce randomness for supporting multi-objective test optimization	[P16]
17	Execute fewer test cases, and detect faults as early as possible	[P17]
18	Generate high quality test suite for combinatorial testing.	[P18]
19	Effective test cases with shortest execution time.	[P19]
20	Reduce the cost of regression testing by identifying and removing redundant test cases from the original test suite.	[P20]
21	Investigate and compare clustering-based approach for inadequate and adequate TSR techniques.	[P21]
22	An intelligent testing approach to remove unessential test cases economically	[P22]
23	This paper aims at reducing the cost of the test by eliminating the redundant test cases.	[P23]
24	Deriving the subset of the original test suite that covers the testing requirements without affecting the accuracy of the test result.	[P24]
25	A tool or framework that can perform Test Case Prioritization (TCP) and Test Case Reduction (TCR).	[P25]
26	Optimizing the test suite without compromising the software quality.	[P26]
27	Selection and execution of necessary test cases.	[P27]
28	Reduce the time spent in testing by reducing the number of test cases.	[P28]
29	Regression tests on database applications under an iterative and incremental development setting	[P29]
30	Address insufficient path coverage or redundant testing in non-cryptographic application.	[P30]
31	Explore fault localization techniques to enhance precision in fault detection.	[P31]
32	Improve fault detection effectiveness by structural profile information.	[P32]
33	Consider regression testing for Information systems that are connected to DBs	[P33]
34	To reduce the test suite size for white box and black box testing generated	[P34]

general by designing approaches for “software testing.” We assume that these TSR models can be applied to any testing suite. A detail description is mentioned in [Table 10](#). The most common among them were the ones developed for reduction of regression tests. Since the size of test suite increases with each increment.

The studies explored clustering algorithm for TSR but these algorithms have their fair share of challenges. The most prominent objectives mentioned in the studies were considering functional aspects of Test suite, consider modification in database while test suite selection, exploring the trade-offs between fault detection and time consumption of for execution of test suite, fault localization, appropriate number of clusters to avoid loss in coverage score. The objectives against each study are mapped in [Table 11](#).

RQ-7: What Were the Challenges Presented in the Field and the Proposed Solutions?

The challenges and their respective solutions provided by authors are compiled in the [Table 12](#).

RQ-8: Which Feature Selection or Feature Reduction Techniques Were Utilised?

The only feature reduction technique mentioned in the articles was Principal Component Analysis (PCA) [P23]. Other studies did not mention the use of feature reduction or feature selection techniques.

The summary of the findings of this study is presented in concise form in [Figure 7](#).

Discussion

General Discussion

To the best of our knowledge, this is the first SMS on unsupervised learning approaches for test suite selection. For this purpose, 254 research papers were retrieved from several databases, of which 34 were used in this study. The largest number of selected studies were retrieved from SCOPUS (i.e., 10 research papers). IEEE Xplore and ACM digital library also contributed 7 research papers each. Most of them focused on test suite reduction, some specifically on regression testing, and one of the studies concentrated on mutation testing. It was also observed that some studies focused on specific testing, such as web application testing and database testing.

The main findings related to some research questions are discussed as follows:

- RQ-1*-Among clustering algorithms, the K-means algorithm is the most widely used one (12 out of 34 studies). One of the studies has also used a variation of this algorithm: K-means++. This was followed by hierarchical clustering, which was used by three studies, and three studies have used it along with agglomerative clustering. Some algorithms like Cobweb and subtractive clustering algorithms were used only in single studies. The names of algorithms used were not mentioned in some research works. Most algorithms were able to perform well by providing reduced test suites with desirable coverage.

- RQ-2*-Coverage and Test suite size reduction are the most preferred metrics of evaluation used in these 34 studies (10 studies each). Coverage metrics included several types of coverage, such as branch coverage, statement coverage, and condition coverage. Multiple condition coverage, path coverage, and modified condition/decision coverage (MC/DC) were also used. Test suite size reduction was also a widely preferred metric since it

Table 12. Challenges and proposed solution mentioned in the articles.

No.	Challenges	Proposed Solution	References
1	The existing solutions produces information that is not linked to the functional aspects of the software.	The approach groups test data into similar functional clusters.	[P1]
2	Due to time constraint, the TSR is mandatory, but the challenge is to maintain the mutation score or code coverage.	Cluster test cases based on similarity and to select a central test case from each of the clusters to be used in the new reduced test suite.	[P2]
3	The challenge during test suite reduction is keeping the cost inexpensive than the execution of whole test suite.	Clustering based on higher frequency and ordering the test cases through similarity score. Then selecting the test cases according to given resources.	[P3]
4	Regression testing for a massive system is expensive, and the evaluation metrics are not available for high scale systems.	Algorithms from the big data domain are utilized, and the similarity between test cases is evaluated to avoid redundancy.	[P4]
5	A solution for higher order HO mutants is needed for self-organizing systems, mainly during re-configuration.	Established clustering algorithms were deployed for TSR and the result was promising.	[P5]
6	Many tests fail due to few underlying faults.	Using clustering algorithms to group the failed test and then analyze only the representative failed test to get the idea about the underlying fault.	[P6]
7	Labeling is expensive and requires manual labor.	Utilize clustering to divide the test inputs, then select the test case which representative of each group. Also include the test cases from minority space. This selected small set is then labeled.	[P7]
8	Minimization of test suit to gain maximum statement coverage in the minimum execution time.	Using clustering algorithms, a subset of test suite is extracted and then using Hitting Set Directed Acyclic Graph (HS-DAG) algorithm amount of time for each test case is calculated.	[P8]
9	In case of multiple faults, it's difficult to determine when to stop fault localization process.	an iterative process of selecting test cases for effective fault localization (IPSETFUL), to identify as many faults as possible in the program until the stopping criterion is satisfied.	[P9]
10	Strength and weaknesses of adequate vs inadequate test suit reduction techniques.	Drawing the comparison on public data set.	[P10]
11	The increase in the number of objectives reduces the performance of the existing techniques. Making them costly and time consuming.	The Resemblance-Based Cluster Head (RBCH) algorithm is proposed to select the Cluster Head (CH) based on the overall similarity between the test cases. The Distance-Based Transposition (DBT) is proposed to prioritize the optimal test case clusters in the distributed environment.	[P11]
12	Getting rid of similar or redundant test cases.	Density-based clustering technique is explored to remove redundant test cases.	[P12]
13	Achieve a reduced of test suit, maximized fault detection rate, and reduced execution time.	adaptive neuro-fuzzy inference system (ANFIS) was devised. Size and execution time of the test suite is reduced up to 50%	[P13]
14	Explore the tradeoff between reduction in test suite size and loss in fault detection capability based on three criteria i.e., statement, method, and class coverages	Clustering based approaches reduced the size and time of execution that the loss in fault detection capability seems acceptable.	[P14]
15	Plugin for eclipse to aid in regression testing by exploiting the inadequate TSR approaches.	A prototype ClUstering-based TEst suite Reduction (CUTER) was developed.	[P15]
16	Existing multi-objective search algorithms have certain randomness when selecting parent solutions for producing offspring solutions.	Proposed a cluster-based genetic algorithm with elitist selection (CBGA-ES) to reduce randomness.	[P16]

(Continued)

Table 12. (Continued).

No.	Challenges	Proposed Solution	References
17	Existing solution does not perform reduction and prioritization simultaneously.	Clustering techniques are used to group test cases with similar fault-detection ability into a cluster, the multi-priority algorithm selects a high-priority test in each cluster.	[P17]
18	The combination test suite generating problem is transformed into a gene sequence optimization problem.	Cluster searching algorithm (CSA), is developed.	[P18]
19	Other solution explored k-means, but it is not cost effective or efficient because it unstable and rarely considers the coverage rates of code.	Cluster the test suite using k-medoids then select the representatives based on code-coverage rate and cyclomatic complexity	[P19]
20	Enhance the multi-objective optimization technique.	K-means algorithm to gather similar test cases in a single cluster. Later, evolutionary algorithm is used to remove redundant test cases, then the test suite is optimized, based on the coverage related criteria, fault-related criteria, and cost-related criteria	[P20]
21	An inadequate approach is appealing when it leads to a greater reduction in test suite size at the expense of a small loss.	Exploring the of trade-offs in test suite reduction between adequate and inadequate TSR approaches using public datasets.	[P21]
22	The test suite after each increment may contain obsolete, redundant, and ambiguous test cases.	The test cases are analyzed to know the difference and similarity value of the test case pairs. Then the greedy and clustering methods are applied to optimize the test cases accordingly. The proposed solution is more effective when the fault detection is prioritized over code coverage	[P22]
23	Test suite reduction by keeping the same code coverage is the challenge.	Test cases are generated randomly. The Procedural Language/Structured Query Language (PL/SQL) tool is used to generate test cases. The SPSS software package is used to apply the K-means Clustering algorithm to reduce the test cases.	[P23]
24	Formation of representative set from the original test suite due to time and resource constraints.	Adaptive Elitism Based Intellect approach (AEBI) using clustering technique to reduce the test suite size.	[P24]
25	A tool or framework that can test all the products in Software Product Line (SPL) is needed.	This paper proposes a hybrid approach which combines K-Means and Principal Component Analysis (PCA) approaches to perform SPL testing.	[P25]
26	Test personnel cannot compromise on the quality of the product and cannot afford to miss any defects.	Hierarchical Divisive Clustering considers all test cases as a single cluster initially then with each iteration they are separated based on the similarity score.	[P26]
27	It is a challenge to optimize test cases through clustering due to its implementation complexity and time-consuming nature.	A Unified Modeling Language profile for Test Suite Optimization (UMLTSO) is proposed for java source code.	[P27]
28	While designing test cases, many test cases are developed that are produced in duplicate	Fuzzy clustering is used for cluster analysis. This aids in finding out redundancy incorporated by test cases.	[P28]
29	Regression testing in database applications, must consider aspects of the database along with the product code for product verification.	For each feature (f _i), a selection of test cases associated with the access to the database are performed; afterward, the similarity matrix is constructed based on the access to the fields of the database. Later, the clustering of test cases is applied and, finally the execution of the clusters containing failed test cases is carried out.	[P29]

(Continued)

Table 12. (Continued).

No.	Challenges	Proposed Solution	References
30	Non-cryptographic programs cover different paths under various sensitive inputs, extending existing tools for identifying information leaks to non-cryptographic applications suffers from either insufficient path coverage or redundant testing.	A dynamic analysis framework named SPIDER, that exploit fuzzing, execution profiling, and clustering for a high path coverage and test suite reduction, and then speeds up the dynamic analysis of side-channel vulnerability detection.	[P30]
31	For fault localization current approach draws the comparison of a successful run test case with a failed run test case.	Program cohesion and program history were used in localization of faulty components	[P31]
32	TSR models considers the number of times that a function or statement is executed but ignores the relations and structural information between function calls	Cluster analysis of three different types of structural profiles	[P32]
33	Regression testing also for software products such as information systems that are usually integrated with or connected to DBs	Utilized DB schema to select the test cases based on modified or new features added to software connected with DB.	[P33]
34	Generation and optimization of white box and black box testing.	Use statistical values to analyze related variables and select variables. Then, clustering reduces test cases while maintaining the same coverage.	[P34]

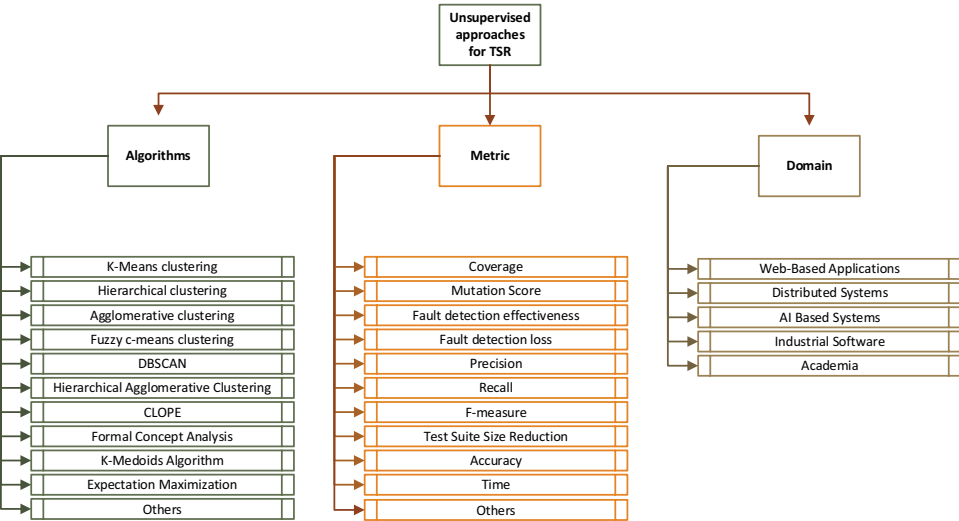


Figure 7. An overview of our SMS.

gives an idea about the size-reducing capability of the method. The least used metrics included mean squared error, effectiveness, and the impact of test suite minimization.

•RQ-3-It was an interesting observation to note that only 3 research papers provided the complete information (code/scripts, framework, and data) required to reproduce their study. 14 of the studies provided details of their pseudocode/novel algorithms in their article. Most of the studies (17 papers) did not make any of their artifacts publicly available to the research

community. Much information related to the data being used in experiments was also not available. However, if they would share them (code/scripts/framework/data), future research would be more reputable and even refutable. We suggest authors share their artifacts in public repositories such as GitHub to motivate new researchers in this field.

- RQ-4*-This was another interesting finding since most of the research papers (28) did not mention anything about the scalability of their approach. Of the remaining 6 research papers, only 3 research papers mentioned that their approach is scalable, 1 of them mentioned that it is not scalable and the remaining 2 required to do more experiments in the future to determine the scalability of their approach. Most studies have mentioned that they are working with large-scale systems but had not mentioned about how scalable their approach is.

- RQ-5*-It was found that the application of TSR using unsupervised learning approaches is not restricted to a specific domain; it has been implemented in real-world applications. Two research articles were based on open-source datasets widely used in academia, but the rest of the research articles tested their models on real-world applications. One of the major drawbacks of datasets originating from academia is that faults are known and manually injected, but real-world bugs could be more catastrophic or unpredictable. Consequently, a model performing well on those datasets might not be able to foresee them and might provide poor performance when deployed. While only some research articles mentioned their domain, the rest did not mention it; however, their methodology and experiment settings provided the idea that they are testing real applications. The domain is not restricted to web-based applications; it ranges from sophisticated embedded systems in vehicles to databases, from cloud-based distributed systems to neural network testing. The potential of this domain is vast and should be fully explored to open new avenues for the future. One study proposed exploring the potential of AI-based testing to fully automate the process (i.e., test generation, selection, and execution processes) (Raamesh and Uma 2011).

- RQ-8*- Only one paper mentioned the use of feature reduction techniques. In ML, the dataset comprising many features can result in sparse data points (Zhou, Ying, and Skiena 2020). Reducing the features or selecting suitable features can result in better ML models (Khoder and Dornaika 2021). Carefully selected features can lead to models with lower training time and storage requirements but with higher performance. This avenue should be explored in future studies for effective results.

This study aimed to provide some valuable insights in the field of using unsupervised machine learning approaches for TSR by finding the commonly used algorithms, metrics, challenges and possible solutions, scalability of the approaches, domains explored, and research objectives. This

will assist researchers in this area in developing new solutions for challenges and identifying unexplored algorithms. The list of metrics will help to choose the best ones for comparing their work. Finally, the list of explored domains will aid in selecting a suitable domain for conducting experiments.

Potential Threats to Validity

The SMS process followed here is vulnerable to threats to validity, including external threats, construct validity, and reliability issues. External validity and construct validity concerns are mitigated by the broad nature of the initial search string and the substantial number of publications (254) retrieved through the initial query search. Reliability concerns have also been addressed, as the process followed in this study has been thoroughly documented, facilitating potential replication in the future. However, it is important to note that variations in judgments may lead to slightly different results.

The selection criteria were analyzed and discussed by the authors to ensure quality assurance. The criteria for research articles were derived from similar studies. However, removing the subjective decisions made by the authors during the scoring of research articles is challenging. The authors engaged in discussions about the review process and assessment criteria in an effort to minimize individual bias. Another challenge involved eliminating papers that included the search string but represented an entirely opposite domain. We mitigated these threats by carefully analyzing and reading the papers. It is important to note that some valuable publications may have been missed during the manual filtering of results from the databases. Although all the synonyms were included in the search query, there is still a possibility of missing some studies that used different synonyms. Notably, this article does not delve into gray literature, such as blogs, websites, technical reports, and white papers on the given topic. Many researchers who share their negative findings are often not published in peer-reviewed journals, yet their insights could be valuable for the given domain. The results presented in this study are based on available literature, and data synthesis was conducted using a well-defined process.

Conclusion and Future Work

After completing the entire SMS process, it was observed that unsupervised learning approaches were effectively employed in the field of test suite reduction/optimization. Most of the studies effectively demonstrate the use of unsupervised learning approaches for test suite reduction. The majority of the studies demonstrated the effectiveness of unsupervised learning algorithms, with a notable reliance on the K-Means algorithm for clustering. The

coverage metric emerged as the most widely used validation metric. The findings indicated that tests within the same cluster often failed for similar reasons. Selecting a subset of tests from different clusters was suggested as a strategy to identify similar number of faults while utilizing fewer testing resources. Additionally, one study recommended using cyclomatic complexity as a hyper parameter for the K-Nearest Neighbor model. Notably, the review highlighted a lack of discussion on scalability in the selected papers. Furthermore, diverse domains were explored by researchers in conducting experiments in this area.

It is important to underscore the industrial significance of our findings. The prevalent use of the K-means algorithm, especially in industrial settings, suggests its practicality and effectiveness. The limited availability of complete information for reproducing experiments is a critical concern for industry professionals. Code, scripts, and data can be stored on platforms such as GitHub, enhancing the transparency of methodologies applied in industrial contexts. The lack of emphasis on scalability in papers highlights a crucial consideration for industries dealing with large-scale systems. From web applications to sophisticated embedded systems, the potential applications of the discussed techniques are vast and extend to domains crucial for industrial processes. In conclusion, our study serves not only as an academic exploration of this field but also as a practical guide for industry professionals. The identified challenges pave the way for the development of robust unsupervised learning approaches tailored to the specific needs of industrial testing scenarios.

For future work, it is necessary to analyze the impact of outliers on a reduced test suite. The DBSCAN clustering algorithm, while effective, is unsuitable for high-dimensional data. Dimensionality reduction techniques should be explored in conjunction with this algorithm. Replicating academic research in an industrial context is essential to validate the effectiveness of developed models. Current models often treat each fault with the same priority, disregarding variations in severity. Future investigations could explore techniques assigning weights to clusters based on fault severity. Exploring the correlation between different metrics can assist developers and engineers in selecting the most appropriate technique for specific environments. Although HDBSCAN algorithm is noise-tolerant, its time-intensive training may hinder efficiency, prompting consideration for more suitable clustering algorithms. To further reduce mutants and test cases, the multi-priority algorithm can be employed alongside the TSR approach. While certain models, such as [P9], have demonstrated effectiveness on subject programs, confidence in their performance on other programs is lacking. Authors, including [P27] and (Zhang et al. 2010), expressed concerns about potential overfitting, suggesting the need for additional empirical studies using different datasets.

It was observed that deep learning (DL) models such as the deep Boltzmann machine have not been utilized for test suite reduction. Researchers are encouraged to explore DL models to enhance test suite reduction. Concerning metrics, using standard metrics in more studies would facilitate result comparisons. In future work, the results of this study can serve as a foundation for addressing other related problems, including test case prioritization and test suite generation.

Acknowledgements

Open Access funding provided by Qatar National Library (QNL).

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

No funding received.

ORCID

Anila Sebastian  <http://orcid.org/0000-0002-9226-3363>

Hira Naseem  <http://orcid.org/0000-0001-6778-0388>

Cagatay Catal  <http://orcid.org/0000-0003-0959-2930>

References

- Ahmad, K., M. Abdelrazek, C. Arora, M. Bano, and J. Grundy. 2023. Requirements engineering for artificial intelligence systems: A systematic mapping study. *Information and Software Technology* 158:107176. doi:[10.1016/j.infsof.2023.107176](https://doi.org/10.1016/j.infsof.2023.107176).
- Ali, N. B., E. Engström, M. Taromirad, M. R. Mousavi, N. M. Minhas, D. Helgesson, and S. Kunze, M. Varshosaz. 2019. On the search for industry-relevant regression testing research. *Empirical Software Engineering* 24 (4):2020–2055. doi:[10.1007/s10664-018-9670-1](https://doi.org/10.1007/s10664-018-9670-1).
- Alkawaz, M. H., and A. Silvarajoo. 2019, December. A survey on test case prioritization and optimization techniques in software regression testing. *Proceeding - 2019 IEEE 7th Conference on Systems, Process and Control, ICSPC 2019*, 59–64. doi:[10.1109/ICSPC47137.2019.9068003](https://doi.org/10.1109/ICSPC47137.2019.9068003).
- Al-Sabbagh, K. W., M. Staron, M. Ochodek, R. Hebig, and W. Meding. 2020. Selective regression testing based on big data: Comparing feature extraction techniques. *Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2020*, 322–29. doi:[10.1109/ICSTW50294.2020.00058](https://doi.org/10.1109/ICSTW50294.2020.00058).
- Amalfitano, D., S. Faralli, J. C. R. Hauck, S. Matalonga, and D. Distanto. 2023. Artificial intelligence applied to software testing: A tertiary study. *ACM Computing Surveys* 56 (3):1–38. doi:[10.1145/3616372](https://doi.org/10.1145/3616372).

- Anwar, Z., and A. Ahsan. 2014. Exploration and analysis of regression test suite optimization. *ACM SIGSOFT Software Engineering Notes* 39 (1):1–5. doi:[10.1145/2557833.2557841](https://doi.org/10.1145/2557833.2557841).
- Azuaje, F., I. Witten, and E. Frank. 2006. Witten IH, Frank E: Data mining: Practical machine learning tools and techniques. *Biomedical Engineering Online* 5 (1):1–2. doi:[10.1186/1475-925X-5-51](https://doi.org/10.1186/1475-925X-5-51).
- Breit, A., L. Waltersdorfer, F. J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, and F. van Harmelen. 2023. *Combining machine learning and semantic web: A systematic mapping study*. New York, NY: ACM Computing Surveys.
- Catal, C. 2011. Software fault prediction: A literature review and Current trends. *Expert Systems with Applications* 38 (4):4626–36. doi:[10.1016/j.eswa.2010.10.024](https://doi.org/10.1016/j.eswa.2010.10.024).
- Catal, C., and D. Mishra. 2013. Test case prioritization: A systematic mapping study. *Software Quality Journal* 21 (3):445–478. doi:[10.1007/s11219-012-9181-z](https://doi.org/10.1007/s11219-012-9181-z).
- Chen, T. Y., and M. F. Lau. 1998. A simulation study on some heuristics for test suite reduction. *Information and Software Technology* 40 (13):777–87. doi:[10.1016/S0950-5849\(98\)00094-9](https://doi.org/10.1016/S0950-5849(98)00094-9).
- Coviello, C., S. Romano, G. Scanniello, A. Marchetto, A. Corazza, and G. Antoniol. 2020. Adequate vs. Inadequate test suite reduction approaches. *Information and Software Technology* 119:106224. doi:[10.1016/j.infsof.2019.106224](https://doi.org/10.1016/j.infsof.2019.106224).
- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39 (1):1–22. doi:[10.1111/J.2517-6161.1977.TB01600.X](https://doi.org/10.1111/J.2517-6161.1977.TB01600.X).
- Dickinson, W., D. Leon, and A. Podgurski. 2001 May. Finding failures by cluster analysis of execution profiles. Proceedings - International Conference on Software Engineering, 2001, 339–48. doi:[10.1109/ICSE.2001.919107](https://doi.org/10.1109/ICSE.2001.919107).
- Felbinger, H., F. Wotawa, and M. Nica - 2017 IEEE International, and undefined 2017. IEEE International, and undefined 2017 Mutation score, coverage, Model inference: Quality assessment for t-Way Combinatorial Test-Suites. Ieeexplore.Ieee.Org, Tokyo, Japan.
- Felbinger, H., F. Wotawa, and M. Nica. 2016 IEEE International, and undefined 2016. n.d. Test-suite reduction does not necessarily require executing the program under test. Ieeexplore.Ieee.Org.
- Ganter, B. and R. Wille. Formal concept analysis - . . . - google scholar. n.d.
- Golagha, M., C. Lehnhoff, A. Pretschner, and H. Ilmberger. 2019. Failure clustering without coverage. Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, 134–145. ISSTA 2019. New York, NY, USA: Association for Computing Machinery. doi:[10.1145/3293882.3330561](https://doi.org/10.1145/3293882.3330561).
- Gupta, N., A. Sharma, and M. K. Pachariya. 2022. Multi-objective test suite optimization for detection and localization of software faults. *Journal of King Saud University-Computer & Information Sciences* 34 (6):2897–909. doi:[10.1016/j.jksuci.2020.01.009](https://doi.org/10.1016/j.jksuci.2020.01.009).
- Habib, A. S., S. U. R. Khan, and E. A. Felix. 2023. A systematic review on search-based test suite reduction: State-of-the-art, taxonomy, and future directions. *IET Software* 17 (2):93–136. doi:[10.1049/sfw2.12104](https://doi.org/10.1049/sfw2.12104).
- Harrold, M. J. 2000. Testing: A Roadmap. Proceedings of the Conference on The Future of Software Engineering, 61–72. ICSE '00. New York, NY, USA: Association for Computing Machinery. doi:[10.1145/336512.336532](https://doi.org/10.1145/336512.336532).
- Harrold, M., J. R. Gupta, and M. Lou Soffa. 1993. A methodology for controlling the size of a test suite. *ACM Transactions on Software Engineering & Methodology (TOSEM)* 2 (3):270–85. doi:[10.1145/152388.152391](https://doi.org/10.1145/152388.152391).
- Holcombe, M., F. Ipate, and A. Grondoudis. 1995. Complete functional testing of safety critical systems. *IFAC Proceedings Volumes* 28 (25):199–204. doi:[10.1016/S1474-6670\(17\)44845-2](https://doi.org/10.1016/S1474-6670(17)44845-2).

- Hsueh, C. H., Y. Pin Cheng, and W. Cheng Pan. 2011. Intrusive test automation with failed test case clustering. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, 89–96. [10.1109/APSEC.2011.31](#).
- Iannino, A., and D. M. John. 1990. Software reliability. In *Advances in Computers*, ed. M. C. Yovits, vol. 30, 85–170. Elsevier. doi:[10.1016/S0065-2458\(08\)60299-5](#).
- Jehan, S., and F. Wotawa. 2023. *An empirical study of greedy test suite minimization techniques using mutation coverage*. New York City, United States: IEEE Access.
- Khalilian, A., and S. Parsa. 2009, October. Bi-criteria test suite reduction by cluster analysis of execution profiles. (LNCS) *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7054:243–56. doi:[10.1007/978-3-642-28038-2_19](#).
- Khan, S. U. R., S. Peck Lee, R. Wasim Ahmad, A. Akhunzada, and V. Chang. 2016. A survey on test suite reduction frameworks and tools. *International Journal of Information Management* 36 (6):963–75. doi:[10.1016/J.IJINFOMGT.2016.05.025](#).
- Khan, K., S. Ur Rehman, K. Aziz, S. S. Simon Fong, and A. Vishwa. 2014. DBSCAN: Past, Present and Future. 5th International Conference on the Applications of Digital Information and Web Technologies, ICADIWT 2014, 232–38. doi:[10.1109/ICADIWT.2014.6814687](#).
- Khoder, A., and F. Dornaika. 2021. Ensemble learning via feature selection and multiple transformed subsets: Application to image classification. *Applied Soft Computing* 113:108006. doi:[10.1016/j.asoc.2021.108006](#).
- Kiran, A., W. Haider Butt, M. Waseem Anwar, F. Azam, and B. Maqbool. 2019. A comprehensive investigation of modern test suite optimization trends, tools and techniques. *Institute of Electrical and Electronics Engineers Access* 7:89093–117. doi:[10.1109/ACCESS.2019.2926384](#).
- Kitchenham, B., and S. Charters. 2007. Guidelines for performing systematic literature reviews in software engineering version 2.3. *Engineering* 45 (4ve):1051.
- Lewis, W. E. 2004. *Software testing and continuous quality improvement*. Boca Raton, FL, USA: Auerbach publications.
- Martins, R., R. Abreu, M. Lopes, and J. Nadkarni. 2021. Supervised learning for test suit selection in continuous integration. *Proceedings - 2021 IEEE 14th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2021*, 239–46. doi:[10.1109/ICSTW52544.2021.00048](#).
- Miller, B. P., L. Fredriksen, and S. Bryan. 1990. An empirical study of the reliability of UNIX utilities. *Communications of the ACM* 33 (12):32–44. doi:[10.1145/96267.96279](#).
- Mottaghi, N., and M. Reza Keyvanpour. 2018. Test suite reduction using data mining techniques: A review article. 18th CSI International Symposium on Computer Science and Software Engineering, CSSE 2017 2018-Janua (March), 61–66. [10.1109/CSICSSE.2017.8320118](#).
- Oded, M. and L. Rokach. 2010. Data Mining and. . . - Google Scholar. n.d.
- On, J. M. n.d. *Proceedings of the fifth Berkeley symposium, and undefined 1967. Some Methods for Classification and Analysis of Multivariate Observations*, Oakland, CA, USA. *Books.Google.Com*.
- Orso, A., and G. Rothermel. 2014 May. Software testing: A research travelogue (2000-2014). *Future of Software Engineering, FOSE 2014 - Proceedings*, 117–32. [10.1145/2593882.2593885](#).
- Petersen, K., R. Feldt, S. Mujtaba, and M. Mattsson. 2008, June. Systematic mapping studies in software engineering. 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12, Bari, Italy, 1–10.
- Raamesh, L., and G. V. Uma. 2011. Method to improve the efficiency of the software by the effective selection of the test cases from test suite using data mining techniques.

- Rebala, G., A. Ravi, and S. Churiwala. 2019. Machine learning definition and basics. *An Introduction to Machine Learning* 1–17. doi:10.1007/978-3-030-15729-6_1.
- Rehman, K., U. Saif, S. P. Lee, N. Javaid, and W. Abdul. 2018. A systematic review on test suite reduction: Approaches, experiment's quality evaluation, and guidelines. *Institute of Electrical and Electronics Engineers Access* 6 (February):11816–41. doi:10.1109/ACCESS.2018.2809600.
- Rothermel, G., R. H. Untch, C. Chu, and M. Jean Harrold. 1999. Test case prioritization: An empirical study. Proceedings IEEE International Conference on Software Maintenance-1999 (ICSM'99). 'Software Maintenance for Business Change' (Cat No. 99CB36360), Oxford, England, 179–88.
- Sampath, S., and R. C. Bryce. 2012. Information and software technology, and undefined 2012. *Information and Software Technology* 54 (7):724–38. doi:10.1016/j.infsof.2012.01.007. Improving the Effectiveness of Test Suite Reduction for User-Session-Based Testing of Web Applications.
- Saraswat, P., A. Singhal, and A. Bansal. 2019. A review of test case prioritization and optimization techniques. *Advances in Intelligent Systems & Computing* 731:507–16. doi:10.1007/978-981-10-8848-3_48/TABLES/3.
- Shahamiri, S. R., W. Mohd Nasir Wan Kadir, and S. Zaiton Mohd-Hashim. 2009. A comparative study on automated software test oracle methods. 2009 Fourth International Conference on Software Engineering Advances, 140–45. doi:10.1109/ICSEA.2009.29.
- Shi, A., A. Gyori, M. Gligoric, A. Zaytsev, and D. Marinov. 2014. Balancing trade-offs in test-suite reduction. Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering 16-21-Nove (November): 246–56. doi:10.1145/2635868.2635921.
- Sun, H., X. Chenchen, and H. Suominen. 2021. Analyzing the granularity and cost of annotation in clinical sequence labeling. <http://arxiv.org/abs/2108.09913>.
- Yager, R. R. 2000. Intelligent control of the hierarchical agglomerative clustering process. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 30 (6):835–45. doi:10.1109/3477.891145.
- Yoo, S., and M. Harman. 2012. Regression testing minimization, selection and prioritization: A survey. *Software Testing Verification and Reliability* 22 (2):67–120. doi:10.1002/stvr.430.
- Zhang, C., Z. Chen, Z. Zhao, S. Yan, J. Zhang, and X. Baowen. 2010. An Improved Regression Test Selection Technique by Clustering Execution Profiles. 2010 10th International Conference on Quality Software, 171–79. doi:10.1109/QSIC.2010.16.
- Zhou, B., Y. Ying, and S. Skiena. 2020. Online AUC optimization for sparse high-dimensional datasets. Proceedings - IEEE International Conference on Data Mining, ICDM 2020-November, 881–90. doi:10.1109/ICDM50108.2020.00097.

Primary studies (sources reviewed in the SMS)

Appendix A. Selected papers in this study

Title	Database	Year of Publication	Reference
Linking software testing results with a machine learning approach	ScienceDirect	2013	[P1]
On Using k-means Clustering for Test Suite Reduction	IEEE	2020	[P2]
Frequent segment clustering of test cases for test suite reduction	Scopus	2014	[P3]
Scalable Approaches for Test Suite Reduction	ACM	2019	[P4]
Test Suite Reduction for Self-organizing Systems: A Mutation-based Approach	ACM	2018	[P5]
Failure clustering without coverage	ACM	2019	[P6]
Practical Accuracy Estimation for Efficient Deep Neural Network Testing	ACM	2020	[P7]
Multi objective test case minimization collaborated with clustering and minimal hitting set	Scopus	2014	[P8]
IPSETFUL: an iterative process of selecting test cases for effective fault localization by exploring concept lattice of program spectra	Springer	2016	[P9]
Adequate vs. inadequate test suite reduction approaches	ScienceDirect	2020	[P10]
Multi-Objective based test case selection and prioritization for distributed cloud environment	ScienceDirect	2021	[P11]
An Efficient Approach for Test Suite Reduction using Density based Clustering Technique	Manual	2014	[P12]
Neuro-fuzzy modeling for multi-objective test suite optimization	Scopus	2016	[P13]
An Empirical Study of Inadequate and Adequate Test Suite Reduction Approaches	ACM	2018	[P14]
CUTER: Clustering-based test suite reduction	ACM	2018	[P15]
CBGA-ES: A Cluster-Based Genetic Algorithm with Elitist Selection for Supporting Multi-Objective Test Optimization	Scopus	2017	[P16]
An Extensive Study on Multi-Priority Algorithm in Test Case Prioritization and Reduction	ACM	2021	[P17]
Global Optimization for Combination Test Suite by Cluster Searching Algorithm	Scopus	2017	[P18]
Test-Suite Reduction Based on K-Medoids Clustering Algorithm	IEEE	2017	[P19]
Test Suite Reduction via Evolutionary Clustering	IEEE	2021	[P20]
Clustering support for inadequate test suite reduction	IEEE	2018	[P21]
A new similarity-based greedy approach for generating effective test suite	Scopus	2018	[P22]
An efficient approach for test suite reduction using K-means clustering	Scopus	2018	[P23]
Test suite reduction based on knowledge reuse: An adaptive elitism based intellect approach (AEBI) using clustering technique	Scopus	2019	[P24]
A hybrid approach to perform test case prioritization and reduction for software product line testing	Scopus	2020	[P25]
Achieving Agility in Projects Through Hierarchical Divisive Clustering Algorithm	Springer	2022	[P26]
A Model-Driven Approach for Simplified Cluster Based Test Suite Optimization of Industrial Systems – An Introduction to UMLTSO	Springer	2019	[P27]
Software testing optimization through test suite reduction using fuzzy clustering	Springer	2013	[P28]
An approach for regression testing of database applications in incremental development settings	IEEE	2017	[P29]
SPIDER: Speeding up Side-Channel Vulnerability Detection via Test Suite Reduction	IEEE	2022	[P30]
Clustering Based Test Suite Selection for Ranking of Program Execution Sequence Using Improved Precision in Regression Testing	Scopus	2019	[P31]
Empirical study of the effects of different profiles on regression test case reduction	Wiley	2015	[P32]
Regression Testing of Database Applications Under an Incremental Software Development Setting	IEEE	2017	[P33]
Test case reduction case study for white box testing and black box testing using data mining.	Manual	2014	[P34]