Received August 2, 2019, accepted August 22, 2019, date of publication August 29, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938260

A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms

ANU BAJAJ AND OM PRAKASH SANGWAN
Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Haryana 125001, India
Corresponding author: Anu Bajaj (er.anubajaj@gmail.com)

ABSTRACT Regression testing is the essential process of software maintenance and evolution phase of the software development life cycle for assuring the quality and reliability of updated software. Test case prioritization is the technique of regression testing to reduce the time and effort required for regression testing. Search-based algorithms are used to enhance the efficiency and effectiveness of the method. Among these search-based optimization algorithms, genetic algorithms are becoming more popular among researchers since the last decade. In this paper, we are doing a systematic literature review, i.e., a secondary study of test case prioritization using genetic algorithms. The objective of this review is to examine and classify the current state of use of the genetic algorithm in test case prioritization. In other words, to give a base for the advancement of test case prioritization research using genetic algorithms. With the use of the systematic literature review protocol, we selected the most relevant studies (20 out of 384) from the appropriate repositories by using a set of search keywords, inclusion/exclusion criteria and the quality assessment of studies. The data extraction and synthesis process and the taxonomic classification are used to answer the research questions. We also performed a rigorous analysis of the techniques by comparing them on research methodology, the prioritization method, dataset specification, test suite size, types of genetic algorithms used, performance metrics, and the validation criteria. The whole process took four months for comprehensive analysis and classification of primary studies. We observed that the parameter settings, the type of operators, the probabilistic rate of operators, and fitness function design have a significant impact on the quality of the solutions obtained. This systematic literature review yields that genetic algorithms have great potential in solving test case prioritization problems, and the area is open for further improvements. Future researchers can fill the research gaps by following the suggestions given in the review. From this review, we found that the use of the appropriate approach can make a genetic algorithm based test case prioritization one of the effective methods in regression testing.

INDEX TERMS Genetic algorithm, NSGA-II, regression testing, systematic review, test case prioritization.

I. INTRODUCTION

The software industry is developing at an unprecedented rate. The software industry needs to be updated or enhanced its product, due to the changing customer(s) requirements or for maintenance to survive in this competitive world. Regression testing plays a very crucial role to ensure the reliability and quality of the software product served in the market. However, it is a very costly and time-consuming process to rerun all the test cases every time a software is updated [1]. Test case prioritization (TCP) helps in sorting only those test cases, which are of an utmost priority by using some test adequacy criteria, e.g., early fault detection [2]. TCP is an NP-Hard problem as it needs to check all the possible permutation sequences [3]. Also, due to the exponential growth of software, we need to resort to meta-heuristics optimization techniques to solve the problem within the stipulated time [4]. Henceforth, the researchers started looking into nature, how nature maintains the optimum balance, and now they are trying to find optimal orderings with the help of nature-inspired algorithms. The researchers have used various Nature-Inspired algorithms in the TCP domain [5]. Inspired by the evolution process, John Holland [6] proposed a genetic algorithm (GA), which became a well-tested

The associate editor coordinating the review of this article and approving it for publication was Hisao Ishibuchi.

meta-heuristic technique. Due to its increasing popularity, the industry has started using GA in software development [7], [8]. GA works on Darwin's theory of evolution. The process begins with the initial population and based on some selection criteria, the parents of the next generations are selected. The selected parents then go through the process of exchanging the genes using the crossover and mutation operators and repeat this process until a termination point [9].

In this paper, we have conducted a systematic review of TCP using GA to know the current status of research in this area. It includes finding strengths and limitations and the future scope of application of GA in TCP to assist the new researchers in this domain. Before going into details, we have given a brief introduction about TCP, GA, and the use of GA in TCP.

A. TEST CASE PRIORITIZATION

Regression testing (retesting) is required to validate the software in the maintenance and evolution phase of the software development life cycle. It is done to ensure that the modifications have not hampered the earlier software versions and the new version is backward compatible. Testing takes around half of the cost of the total software development cost [10]. It is a very time consuming and expensive process. We can optimize the regression testing process in three ways: 1) Test Suite Minimization (TSM) which minimizes the test suites by deleting the obsolete test cases. 2) Test Case Selection (TCS) selects only those test cases that are related to some specific criteria, e.g., cover an updated area only. 3) Test Case Prioritization orders the test cases on some properties so that highly ranked test cases execute at a higher priority [1]. Both TSM and TCS truncates the original test suites, while the TCP only reorders the test suites without removing any test cases. Sometimes there are test cases that are not required at a particular version but can be useful in later versions of a software [11]. In other words, prioritization is safer than permanent removal [5], i.e., TCP is a secure, reliable, and cost-effective method for regression testing. Therefore, researchers focus is more on TCP than the test suite minimization or test case selection [1].

The researchers broadly categorized the TCP methods into code-coverage based, requirements-based, history-based, fault-based and search-based techniques [12] (For details see Section IV-B). When we have a source code available, then the prioritization which ranks the test cases based on statement/block/method coverage is known as code coverage based prioritization [13]. Requirement based prioritization utilizes the customer's requirements to order the test cases. The fault coverage based method prioritizes the test cases based on the rate of faults covered [14]. Another technique is history-based prioritization, which uses historical information about software [11]. Search-based prioritization helps in finding the optimum ordering of the test cases by searching over the global space for single or multiple objectives. Researchers have used various search-based techniques for TCP, e.g., greedy algorithm [15] and hill-climbing [16]. In this paper, we have focused only on GA based TCP.

B. GENETIC ALGORITHMS

The problems are getting more and more complex, and it is becoming challenging to solve the problems with deterministic algorithms [17]. Influenced by the evolutionary process of natural species, various evolutionary algorithms are proposed to solve NP-hard problems [18]. Genetic Algorithm is the one which became popular in the early 1970s when the book by John Holland, "Adaption in Natural and Artificial Systems" [6], came into the market. Darwin's theory of 'survival of the fittest' [16] formulates the basis for the development of GA. The underlying principle is that individuals in the population fight for resources. The individuals who succeed produce more offspring, and their genes propagate to the subsequent generations. The chromosomes of the parents mate together by exchanging or altering the genes to produce offspring, which may have better fitness than either of the parents. This process keeps on going for generations to form better individuals of the species to adapt to the environment [19].

C. GENETIC ALGORITHMS IN TEST CASE PRIORITIZATION

The TCP problem can be mapped on to GA by representing the genetic information, i.e., chromosomes as a sequence of test cases [20]. The problem is discrete as it is related to the ordering of test cases. Hence, the chromosome is represented using permutation encoding, which assigns a sequence number to each test case [21].

The process starts by forming the initial population of random individuals. The population undergoes the fitness test, which is calculated using some coverage criteria to prioritize the test suites. The selection mechanism, e.g., roulette wheel selection, truncation selection, and tournament selection, choose the test suites whose fitness is higher than others to form the next generation. The selected test suites join together in pairs by using crossover and mutation operators to create better offspring for the next generation.

The crossover operator is just the recombination of two chromosomes which inherits the characteristics of both the parents [11]. TCP uses crossover operators like an ordered crossover, and partially matched crossover (PMX). Fig. 1 shows the process of ordered crossover. In the ordered crossover, we randomly select two positions to say 3 and 5 in this example. Copy the test cases of one parent at the defined locations and arrange the leftover test cases of another parent in the same order of their occurrence to produce the new offspring. Whereas, PMX crossover does the position by position exchanges instead of just sliding motion [22].

The mutation operator is used to maintain the diversity of individuals in the new generation [11]. Fig. 2 shows the most commonly used mutation operator, i.e., swap mutation. In swap mutation, we randomly select two positions to say, 3 and 6 in the figure and swap the corresponding test cases.

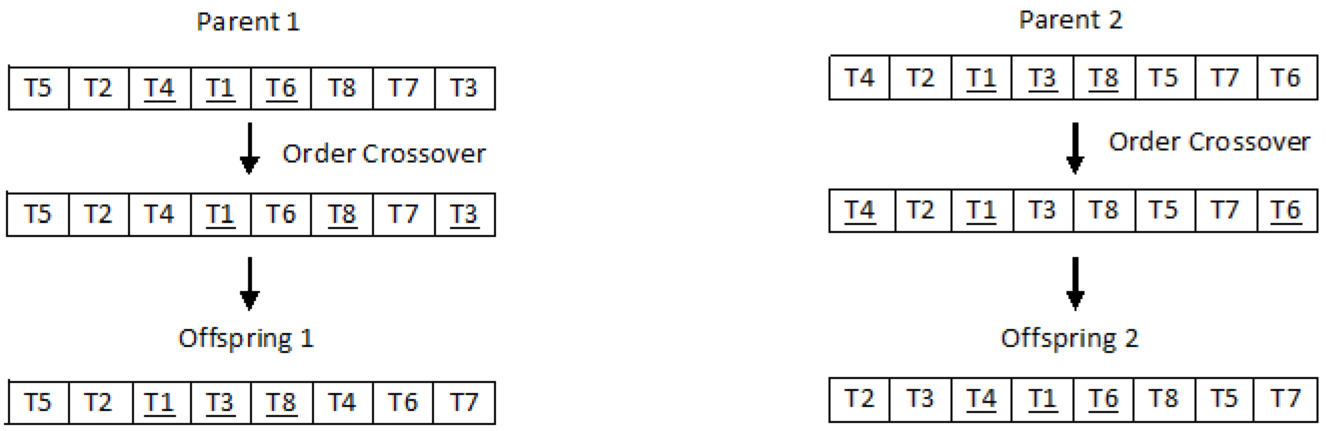

FIGURE 1. **Ordered crossover.**

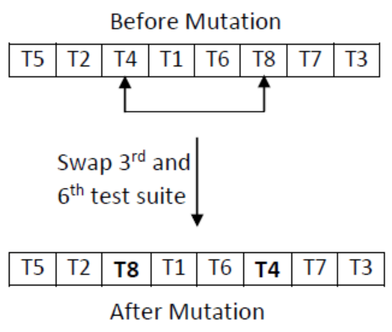

FIGURE 2. **Swap Mutation.**

This process continues to a certain termination point [9]. In this way, GA helps in finding the optimal ordering of test cases. It uses genetic information of chromosomes to guide the search for locating the best solutions in the search space [16].

Various researchers have applied genetic algorithms in TCP, and we have tried to find out the potential of the same. We can broadly classify the GA into two classes, i.e., single-objective GA and multi-objective GA. The single objective genetic algorithms are simple GA or some enhanced/improved/modified version of GA as per need of research work. Whereas, multi-objective GA (MOGA) considers two or more objectives at a time for prioritizing the test cases. A brief explanation of MOGAs used in TCP problem is as follows:

NSGA-II: Deb *et al.* [23], proposed Non-Dominated Sorting Genetic Algorithm-II (NSGA-II). It is the extension of GA for multi-objective optimization. It sorts the population according to the objectives which are non-dominating in nature. It also preserves elitism (keep the best individuals from the parent and child population) and the diversity of solutions. There are three versions of NSGA: Original NSGA, NSGA-II, and NSGA-III.

WBGA: Weight-Based Genetic Algorithm (WBGA) converts the multi-objective problem into a single-objective one by assigning weights to each objective [7]. For example, Random Weight Based Genetic Algorithm (RWGA) dynamically assigns weights to each objective by exploring the search space during each generation.

SPEA2: Strength Pareto Evolutionary Algorithms (SPEA2) is also the multi-objective extension of GA. It uses the external archive, to store non-dominated solutions. If an archive overflows, then it deletes the solutions utilizing the clustering mechanism. The strength values of each individual are summed up to get fitness assignments. The binary tournament selection scheme selects the individuals from the archive and population members for the mating selection. Apply recombination operator and mutation operator to produce a new population [24].

MoCell: Nebro *et al.* [25], developed a Multi-Objective Cellular Genetic Algorithm (MoCell), which stores the non-dominated solution in an external archive. In each iteration, it uses the feedback mechanism to randomly replace the existing individuals of the population from the archive.

Other MOGAs: Some other MOGAs are Pareto Archived Evolutionary Strategy (PAES) [26] and Pareto Envelope Based Selection Algorithm (PESA-II) [27]. Researchers have used some other approaches along with GA, e.g., Pradhan *et al.* [28], proposed clustering-based GA (CBGA-ES) for TCP, i.e., they divided the population into clusters to group solutions of similar quality. Cluster dominance strategy was used to find the solutions from the best cluster for next-generation production. For a better understanding of the researchers', Fig. 3 shows a taxonomic classification of GAs used in TCP.

II. RELATED WORK

There are various nature-inspired approaches for TCP in the literature, but GAs are gaining more popularity. Due to its successful working in numerous applications, it has become a state-of-the-art algorithm. The systematic review study can

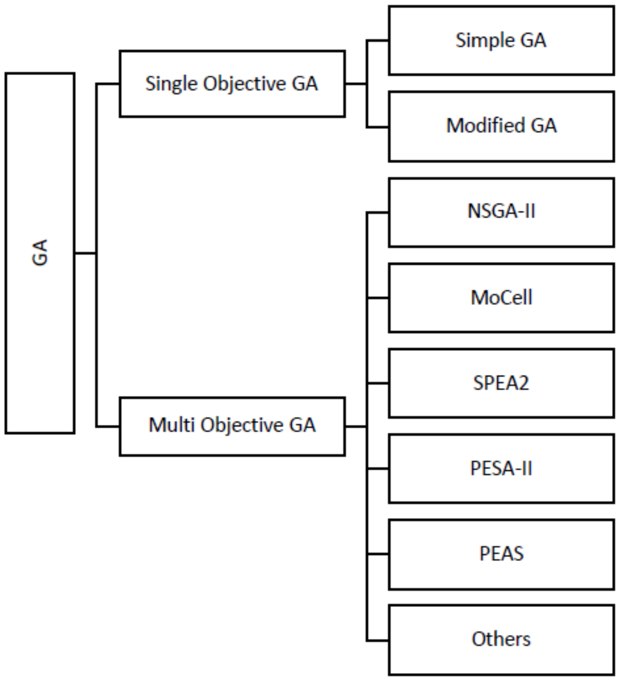

FIGURE 3. Taxonomic classification of GAs used in TCP.

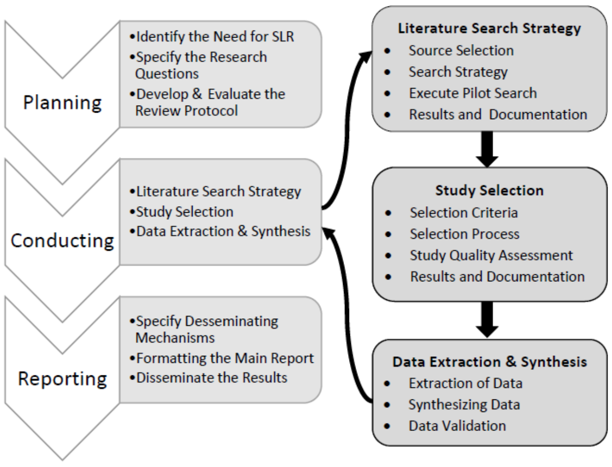
Planning
•Identify the Need for SLR
•Specify the Research Questions
•Develop & Evaluate the Review Protocol
Conducting
•Literature Search Strategy
•Study Selection
•Data Extraction & Synthesis
Reporting
•Specify Desseminating Mechanisms
•Formatting the Main Report
•Disseminate the Results
Literature Search Strategy
Source Selection
Search Strategy
Execute Pilot Search
Results and Documentation
Study Selection
Selection Criteria
Selection Process
Study Quality Assessment
Results and Documentation
Data Extraction & Synthesis
Extraction of Data
Synthesizing Data
Data Validation

FIGURE 4. Systematic literature review process.

evaluate the growth of GA in TCP and provide a baseline to future researchers. There are only a few secondary studies, i.e., systematic reviews available on the use of GA in TCP. First of all, Yoo and Harman [1] conducted a systematic survey of regression testing techniques, i.e., test suite minimization, test case selection, and test case prioritization. They discussed four approaches of TCP techniques, i.e., coverage-based, history-based, requirements-based, and model-based approaches and gave an overview of the type of metrics used in TCP. They also analyzed that the trend was shifting more towards test case prioritization instead of the other two. Researchers should take care of multiple concerns like cost and value trade-offs while applying GA in TCP. They covered only two studies that used GA in TCP [13], [16].

Singh *et al.* [12], gave a review specifically on TCP. They categorized TCP techniques into eight approaches, viz., fault-based, coverage-based, requirements-based, modification based, genetic-based, history-based, composite approaches, and other than these. They also indicated the research gaps regarding the use of metrics, tools, and datasets. They discussed only two studies using GA for TCP [13], [29] and concluded that it is a topic of interest for new researchers. Khatibsyarbini *et al.* [30], TCP techniques into twelve classes and did a trend analysis of each class. They also discussed the research gaps regarding the datasets used and why and where to use the evaluation metrics. Whereas Catal and Mishra [31] provided a systematic mapping of TCP techniques. They provided the statistics of the trend of TCP, the most frequently used TCP approaches, research methodology, evaluation metrics, and datasets. All the above researchers conducted secondary studies to find empirical evidence regarding TCP. However, none of them specifically studied the application of GA in TCP. Our work is different from the above researchers because our focus is on getting evidence regarding the application of GA in TCP.

We found only one systematic literature review [32] to date, which discussed the use of GA in TCP. This paper covered seven studies and gave a general description of the use of GA in TCP. Our work is different from this work because we have included more studies. Also, we have performed a rigorous analysis of designing GA for TCP in context to the parameter settings, the fitness function, evaluation metrics, and datasets used. Along with these, we have also provided significant advantages, limitations, and suggestions that may be helpful for future researchers to fill the research gaps.

III. RESEARCH METHODOLOGY

Unlike conventional reviews, Systematic Literature Review (SLR) is a systematic process to collect and summarize the empirical results obtained from the existing literature. In other words, it is a trustworthy, auditable, and rigorous process to know the current status of research in a particular domain. We followed the guidelines of Kitchenhamm and Charters [33] and performed the SLR in three stages: planning, conducting, and reporting (see Fig. 4).

A. PLANNING THE SYSTEMATIC REVIEW

It includes the need for review, specification of research questions, definition, and evaluation of review protocol.

1) NEED FOR SLR

While the progression of research, the preliminary literature review showed that to the best of our information, no SLR had been published so far on the topic of TCP using GA. Considering the increasing use of GA in TCP [5], we believe that it is high time to consolidate and synthesize existing research evidence from the relevant studies. Moreover, we have also classified primary studies into seven classes, which assists in formulating the basis for answering the research questions. In other words, it helps in identifying the key concepts and issues through an in-depth analysis of the current work.

2) DEFINE AND EVALUATE REVIEW PROTOCOL

We have developed a review protocol to guide the execution of SLR and to lessen the researcher bias. It consists of research questions, search strings, repositories to be searched, and the selection criteria, assessment of quality, data extraction, and synthesis, respectively. The external reviewer has evaluated and validated the review protocol and, as per their suggestions, we have incorporated changes to refine the protocol. The subsequent sections provide a comprehensive explanation of each step.

3) RESEARCH QUESTIONS

We have identified five research questions and twelve sub-questions based on our motive to perform the SLR. In other words, the answers to these questions give us the evidence-based consolidation to define, apply, and incorporate GA in TCP. Table 1 presents the research questions and the purpose behind them. Along with the defined research questions, we have also clarified the scope and objective of the review with the help of PICOC (Population, Intervention, Comparison, Outcomes, and Context) criteria (see Table 2).

B. CONDUCTING THE SLR

It performs the search on electronic databases using the search string and extracts the relevant studies to answer the research questions. It includes a literature search strategy, study selection, and, data extraction and data synthesis.

1) LITERATURE SEARCH STRATEGIES

The search process consists of a selection of repositories, the definition of the search string, and execution of a pilot search, refinement of the search string and retrieval of primary studies from the repositories (see Fig. 4).

2) SOURCE SELECTION

We have selected the largest, the most popular online repositories as the literature source, i.e., ACM digital library, IEEE Xplore, Science Direct (Elsevier only), SpringerLink. In addition to these, other sources of evidence, i.e., reference lists and citations of included studies have also been manually searched for ensuring the thoroughness.

3) SEARCH STRATEGY

We have devised a two-stage search strategy comprised of the primary search and the secondary search to ensure the selection of all relevant studies.

Primary search: We have developed the search string (((regression OR software) AND test AND (case OR suite) AND (prioritization OR optimization) AND (“genetic algorithm” OR “genetic programming” OR “search algorithm” OR “meta-heuristics” OR “multi-objective”))) following the suggestions of Buckley *et al.* [34] (see Table 3).

TABLE 1. Research questions and their motivations.

No.	Research Questions	Motivations
RQ1	What is the research trend of GA applications in TCP techniques?	
RQ1.1	How many studies were published over the years and most targeted sources?	To investigate the development of GA based TCP and mostly used coverage criteria to know the current state of research and future possibilities.
RQ1.2	For what prioritization method (coverage criteria), GA has been used in TCP?	
RQ2	What kind of testing environment has been used in GA based TCP?	
RQ2.1	What type of datasets and their granularity has been used?	To investigate the dataset used, the effect of dataset size, test suite size, and the tools required for research.
RQ2.2	Which tools are used?	
RQ3	How GA has been mapped to TCP?	
RQ3.1	Which type of GA is used?	To investigate the type of GA used, the effect of parameters, population size, generation size, population representation on performance, and design of fitness function which illustrates the basic process of GA usage in TCP that may help the researchers in choosing the appropriate settings.
RQ3.2	What kind of parameter settings are there?	
RQ3.3	How researchers designed the fitness function?	
RQ4	How researchers checked the validity/effectiveness of algorithms?	
RQ4.1	Which performance metrics has been used for evaluation?	To find out the popular performance metrics and to investigate the significance of the method. To collect pieces of evidence that GAs performed better than other prioritization techniques that can influence future researchers to work in this area.
RQ4.2	Whether the technique has been compared with existing techniques?	
RQ4.3	Did the research work prove statistically?	
RQ5	What is the significance of using GA in TCP?	
RQ5.1	What are the purpose and limitations of the current research?	To investigate why and where GA has been applied in TCP. To identify possible opportunities and shortcomings of using GA in TCP and provide recommendations (if any).
RQ5.2	What are key issues and gaps that can be addressed in the future for using GA in TCP?	

TABLE 2. PICOC criteria to define the scope and goal of SLR.

Population	The literature on the application of GA in the TCP
Intervention	Taxonomic classification of GA application in TCP.
Comparison	A holistic comparison among intervention, to analyze the cumulative effect of current research on methods and solutions.
Outcomes	Classification framework and the suggestions with the synthesized pieces of evidences to guide and practice research for the application of GA in TCP.
Context	A systematic investigation to consolidate the peer-reviewed search.

Secondary Search: We have used the snowballing procedure [35], [36] for conducting the secondary search. It includes (a) Backward snowballing, i.e., to check the bibliographic information of selected studies to get other appropriate reviews. (b) Forward snowballing, which means to check the citations of chosen studies for any other relevant studies. (c) Identifying and contacting selected studies’ authors to get extended versions of work (if needed). Repeat the process until we did not get any new studies.

4) EXECUTION OF PILOT SEARCH

We did the pilot search of four papers on the ACM Digital Library and IEEE Xplore online resources. We have validated

TABLE 3. Primary search strategy.

Derive search keywords	From research questions and PICOC criteria.
Identify synonyms and alternate spellings of search keywords	For example, test and testing; test case and test suite; prioritization and optimization; genetic algorithm and search algorithm, etc.
Search keyword connectors	1) OR operator considers alternate synonyms and spellings. 2) AND operator links the major search keywords. 3) ONEAR operator means the search keywords on either side of the operator must both be near each other in the text and also appear in the order we have supplied.
Construct sophisticated search strings	Using identified search keywords, Boolean connectors ORs, ANDs, and ONEAR for the different repositories search.

TABLE 4. Repositories and their corresponding search strings.

IEEE	(((regression OR software) AND test AND (case OR suite) AND (prioritization OR optimization) AND ("genetic algorithm" OR "genetic programming" OR "search algorithm" OR "meta-heuristics" OR "multi objective")))
Springer	((regression OR software) ONEAR (test) ONEAR (case OR suite) ONEAR (prioritization OR optimization) AND ("genetic algorithm" OR "genetic programming" OR "search algorithm" OR "meta-heuristics" OR "multi objective"))
Elsevier	Title, abstract, keywords: ((regression OR software) AND ("genetic algorithm" OR "genetic programming" OR "multi-objective" OR "search algorithm" OR "meta-heuristics")) Title: (test OR testing) AND (case OR suite) AND (prioritization OR optimization)
ACM	(((regression OR software) AND test AND (case OR suite) AND (prioritization OR optimization) AND ("genetic algorithm" OR "genetic programming" OR "search algorithm" OR "meta- heuristics" OR "multi objective")))

the inclusivity of the search terms by fetching the sample papers during the testing process. The number of results obtained through the pilot searches worked as an indicator to find out the efforts required for the complete search process. Additionally, it has been used to generate a schedule (See section III-F) to predict how long it will take to complete the entire SLR process.

5) SEARCH RESULTS AND DOCUMENTATION

We have applied four search strings on four different databases (see table 4), and extracted the literature from the year 1999 (no results before it) to August 31, 2018. We have searched by title, abstract, keywords, and bibliographic information in the research papers, and retrieved a total of 384 studies from the repositories. This broad list may include many irrelevant studies. Search results and any modifications made to the search strategy has been well documented and justified (if required).

C. STUDY SELECTION

The selection of primary studies includes the description of selection criteria, the selection process, and the study quality assessment.

TABLE 5. Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Studies applying GA in TCP.	Studies not related to GA based TCP.
Papers that involve an empirical study or a comparative study of TCP that involved the use of GA.	Studies regarding the proposal of an approach, describing the applications of GA in TCP or review based studies
Studies related to the parameter setting and operators' role of GA in TCP.	Technical reports, government reports, letters and editorials, short notes, abstracts only studies.
Only the latest version of the studies have been included if there is any revision of the papers.	Duplicate studies have been removed.
The time of publication is not a barrier for inclusion.	Non-English studies.

1) SELECTION CRITERIA

This step is required to select only relevant studies and removing irrelative studies. To do this, we use the inclusion and exclusion criteria as defined in Table 5. The first author selected the primary studies and verified the selection process by using a test/retest approach. Additionally, the second author (Ph.D. supervisor) compared the results obtained from the random sample of archived results. In this way, we performed the testing and verification of the appropriateness of inclusion and exclusion criteria as per the suggestion of Brereton *et al.* [37].

2) SELECTION PROCESS

The study selection process consists of two steps: *(a) Initial Selection:* Screening of studies based on titles and abstracts to exclude the irrelevant literature. Followed by the full text read to ensure that selected studies include the relevant information and data to be extracted for later analysis. *(b) Final Selection:* Studies finally chosen by the initial selection went through the secondary search process, i.e., review of the references and the citations of selected studies for further addition.

3) STUDY QUALITY ASSESSMENT

Only inclusion and exclusion criteria are not sufficient enough to extract the most relevant research studies. The quality of the papers has been evaluated based on the following questions:
Q1 Were the research purpose and objectives stated clearly?
Q2 Was the genetic algorithm explained clearly?
Q3 Was the experiment conducted on industrial applications or benchmark programs?
Q4 Did the researchers explicitly defined the used measures?
Q5 Did the researchers validate the results adequately?

We scored the questions on a scale of 0 (No), 0.5 (partial), and 1 (Yes), and their summation yields the quality of the paper. We have set the threshold score to 2.5 to ensure the quality of primary studies. In other words, we have included only those studies whose quality score is more than or equal to 2.5.

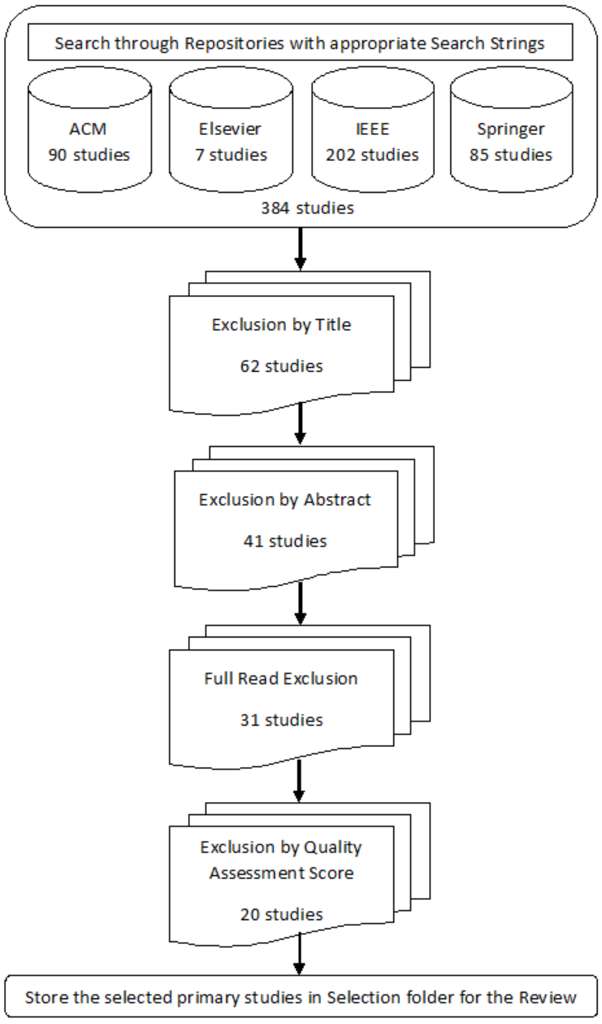

FIGURE 5. Search and selection procedure.

4) SELECTION RESULTS AND DOCUMENTATION

The 384 studies were undergone an initial selection phase and resulted in 31 full-text studies. In the final selection phase, we have covered the grey literature by using snowballing procedure and contacting the authors of primary studies for any recent advancement in the selected papers. However, we did not get any relevant studies from grey literature to include in our SLR. Consequently, we have analyzed the quality of a total of 31 studies, and finally, we have selected 20 papers for this review. Table 6 presents the quality assessment score (QAS) of the selected studies along with their purpose. We have saved suitable studies along with a copy of the abstract in the separate folder to make the review transparent and repeatable. Henceforth, we have assembled a concluding list of 20 primary studies which included studies obtained through search in the repositories, snowballing, and quality assessment scores for relevance. Fig. 5 shows the complete process of the search and selection of primary studies.

TABLE 6. Purpose and quality assessment score of the selected studies.

ID	Work	Purpose	QAS
P1	Walcott and Kapfhammer (2006) [13]	To develop time constraint TCP based on coverage information using different parameter settings. To calculate space and time overhead calculations.	4
P2	Li *et al.* (2007) [16]	To find out the factors affecting the efficiency of the algorithm and effective fitness metric.	5
P3	Wang *et al.* (2014) [8]	To propose search based TCP for software product line testing by studying multiple cost-effectiveness measures. To check the performance of the algorithm by increasing the number of test resources and features.	4.5
P4	Wang *et al.* (2016) [7]	In addition to, test cases optimal orderings, test case execution optimally allocate required test resources in minimum time.	4.5
P5	Epitropakis *et al.* (2015) [44]	To check the effectiveness of the algorithms on three criteria, i.e., code coverage, changed code coverage, and past fault coverage using a lossless coverage compaction algorithm.	5
P6	Li *et al.* (2013) [46]	To enhance the execution efficiency of the algorithm by introducing parallelism in GA.	4.5
P7	Bian *et al.* (2015) [21]	To check the effect of changed code information along with execution time on TCP as compared to all statement coverage for multi-objective TCP.	4
P8	Yuan *et al.* (2015) [3]	To apply epistatic domain theory on crossover operator in GA, to analyze the interactions between genes in GA, and to define the epistatic test case segment for deterministic effect on fitness.	3.5
P9	Pradhan *et al.* (2016) [40]	To propose search based TCP with incremental code coverage and to have a high impact position.	5
P10	Pradhan *et al.* (2017) [28]	To deal with sub-optimal solutions, to add an elitist strategy to reduce the randomness and to find the best solutions from the best cluster by applying cluster dominance strategy.	4.5
P11	Arrieta *et al.* (2016) [47]	To develop search based TCP for configurable cyber-physical systems that detect faults in minimum time and to check the efficiency of the algorithms as the test suite size grows.	4.5
P12	Arrieta *et al.* (2018) [41]	To generate and prioritize reactive test cases for cyber-physical systems by tracing requirements and empirical evaluation of the algorithm with different crossover and mutation rates.	4.5
P13	Masri and El-Ghali (2009) [38]	To propose the TCP technique which takes into account the combinations of profiles of different types and using GA to reduce the cost of identifying these combinations.	3
P14	Conrad *et al.* (2010) [29]	Empirically analyzing the effect of different parameter settings and using an automatic tree model for identifying the best configuration of GA for TCP.	4
P15	Huang *et al.* (2012) [11]	To propose effective cost cognizant version specific TCP that does not consider the uniformity of test cost, the fault severity, and do not require the source code. To study the impact of population size and generation size on the performance of the algorithm.	4.5
P16	Ahmed *et al.* (2012) [43]	To propose multi-criteria fitness function in TCP based on multiple control flow coverage metrics, the number of faults revealed, and their severity.	2.5
P17	Marchetto *et al.* (2016) [45]	To propose multi-objective TCP by applying a metric-based approach that automatically recovers links by identifying the fault-prone areas.	5
P18	Parejo *et al.* (2016) [42]	To propose TCP for highly configurable systems utilizing the functional objectives, nonfunctional objectives, and checking their effectiveness using the fault detection rate.	5
P19	Khanna *et al.* (2018) [39]	To propose TCP for websites in the multi-objective environment, i.e., to consider fault detection capability, fault severity, and execution cost.	3
P20	Di Nucci *et al.* (2018) [15]	To propose multi-objective and many-objective TCP and to assess the selection pressure of hypervolume indicator based GA if the number of objectives increases.	5

D. DATA EXTRACTION AND SYNTHESIS

It comprises the extraction of data from selected primary studies, which is succeeded by analysis of the extracted data and capturing useful information for answering the research questions.

1) DATA EXTRACTION

We have designed a data extraction form (see Table 7) for collecting data from each primary study. The bibliographic information consisted of the following properties. 'Unique ID' for unique identification of paper. Properties 'title' and 'authors' for paper description. Properties 'Publisher', 'publication type', and 'year' to analyze the publication trend and distribution of primary studies over the years. Properties 'email', 'citation count', and 'reference count' for the secondary search to contact the corresponding authors, and searching through citations and references. The research focus tells about the 'purpose' and 'objective' of the study, and the applied 'research methodology' (values: development, experiment, empirical study, or a case study). These above properties described studies, while the rest of the properties were recognized from research questions and the exploration we want to do. The 'prioritization method' contains information regarding the type of coverage criteria used for TCP, i.e., whether it is code coverage, fault coverage, requirement coverage, configuration coverage, or other than this coverage. The category GA information gathered all the GA related data, i.e., the 'type of GA' used, the number of 'executions', 'parameter settings', and 'operator selections', 'fitness function design'. The category dataset specification contained the 'type', 'name', and 'size' of the dataset and 'test suite size' used by researchers. Next, the evaluation mechanism used is reported by properties 'performance metrics' used, how the algorithm is validated, i.e., 'validation criteria', and whether performance significance is proved using a 'statistical test'. Last but not least, the results category contained 'major findings', 'limitations', and 'future work' properties.

2) DATA SYNTHESIS

The data synthesis process converts the extracted data into useful information to answer the research questions. To ease the analysis process, we have designed a structured format and classified all the selected primary studies into seven classes. We further divided each class into sub-classes. The sub-sections of the results' section explain each sub-class. Table 8 presents the description of the class along with its sub-classes. This classification framework worked as a foundation for a comprehensive analysis of research to answer research questions. Table 9 presents the corresponding mapping of the classes with that of primary studies.

3) DATA VALIDATION

The first author extracted all the data, and the second author (the Ph.D. supervisor) independently looked out in the random sample for the data collection [37]. We compared the results and resolved the issues (if any) by discussions. To summarize the results, we need to establish a valid list of properties' values [33]. As it is difficult to predict the values before SLR, therefore we can get a correct list of values once we collected the data from the data extraction process. For example, the dataset size ranges from zero to millions of lines of code. Therefore, we have divided this range into sub-ranges, i.e., small program size (0–10KLOC), medium program size (11-30KLOC), large program size (31-50KLOC), and very large program size (>50KLOC). We followed the same procedure for test suite size, i.e., each primary study considered how many test cases for prioritization. Where we have small test suite size (0-100 test cases), medium test suite size (101-500 test cases), large test suite size (501 to 3000 test cases) and very large test suite size (more than 3000 test cases). As all the studies would not fit into the preset values so, we have added one new value 'others'. It supports the studies that rarely used one of the predefined values. To enable data synthesis, we have validated the values of the properties by collecting the valid value set from the random sample of primary studies [33]. We have not modified the set of values after validation.

TABLE 7. Data extraction form.

Category	Sub-Category	Paper Description
Bibliographic Information	Unique ID Title Year Publisher Paper type Authors Corresponding email address Citation count References	
Research Focus	Purpose Main Objectives Methodology	
Prioritization Method	Code coverage, fault coverage, requirement coverage, configuration coverage or others	
GA Information	Type of GA Execution **Parameter Settings and Operators Selection** Encoding Selection Crossover Crossover Rate Mutation Mutation Rate Population Size Generations Size Any other Special Parameters **Fitness Function Design**	
Dataset Specification	Dataset Type Dataset Name Dataset Size Test Suite Size Tools	
Evaluation	Evaluation Metrics Validation Criteria Statistical Test	
Results	Findings Limitations Future Work	

TABLE 8. Classification of papers.

Category No.	Category Type	Description	Sub-categories
C1	Research Methodology	This category tells about the type of research work done by researchers, i.e., whether they developed a new prioritization method or done an experiment, empirically analyze the prioritization approaches or application on some case study.	1.Development 2.Empirical study 3.Experimental Study 4.Case study
C2	Prioritization method	This category tells about the type of testing criteria used for prioritization, i.e., whether they cover the source code, fault, requirements, or configuration of the software under test.	1.Code Coverage 2.Fault coverage 3.Requirement coverage 4.Configuration coverage 5.Others
C3	Type of GA used	This category deals with the type of GA applied for prioritization, i.e., whether it is simple GA, improved/enhanced/modified GA, NSGA-II, and other MOGAs like RWGA and MoCell and SPEA2.	1.Simple GA 2.Modified GA 3.NSGA-II 4.Others
C4	Dataset Specification	This category tells about the type of datasets used for prioritization, i.e., whether it is industrial, laboratory, or open source programs. It also tells about the size of the dataset, whether prioritization is done on the small size (0-10KLOC), medium size (11-30KLOC), large size (31-50KLOC), or very large size programs (greater than 50KLOC).	1.Industrial study 2.Laboratory programs 3.Open Source 4.Small programs 5.Medium programs 6.Large programs 7.Very large programs
C5	Test suite size	This category tells about the number of test cases prioritized by the prioritization. It is divided into four sub-categories where, small test suite represents test cases range from 0-100, medium test suite ranges from 101-500, large test suite ranges from 501-3000, and very large test suite contains greater than 3000 test cases.	1.Small test suite size 2.Medium test suite size 3.Large test suite size 4.Very large test suite size
C6	Performance Metrics	This category tells about the evaluation metrics used in the primary studies, i.e., Average Percentage of Fault Detection (APFD) or its modified versions, Fault Detection Capability (FDC), Execution Time (ET), Coverage Effectiveness (CE), Average Percentage of Element Coverage (APEC) where Element represents the statement, block, or condition, and some other evaluation metrics.	1.APFD 2.FDC 3.APEC 4.ET 5.CE 6.Others
C7	Validation Criteria	This category tells about the effectiveness of the studies, i.e., whether the study is compared with baseline approaches or some state of the art techniques, it is statistically approved or not, whether the studies perform parameter or operators' role efficiency check or proved it statistically.	1.Sanity check 2.Statistical comparison 3.Parameter efficiency check 4.Statistically parameter check

TABLE 9. Mapping table.

ID	C1	C2	C3	C4	C5	C6	C7
P1	1	1	2	3,4	1	1,4	3
P2	2	1	1	2,3,5	4	3	2,4
P3	4	2,5	1	1,2,7	2	2,4,6	2,4
P4	4	2,5	4	1,2,7	3	2,4,6	2
P5	2	1,2	3	3,7	3	1,6	2,4
P6	1	1	2	3,5	4	3,4	3
P7	3	1	3	3,5	4	3,4	4
P8	3	1	2	3,6	3	3	3
P9	4	2,4	3	1,7	2	2,3,6	2
P10	4	2,4	4	1,7	2	2,6	2
P11	4	2	4	1,5	2	2,4	2,4
P12	4	3	3	1,5	1	4,6	2,4
P13	3	1	1	3,4	3	6	1
P14	2	3	1	2,4	1	4,5	4
P15	1	2	1	3,5	2	1,4	1,3
P16	3	1,2	1	2,4	1	1	1
P17	1	1,3	3	2,3,7	3	1,6	2,4
P18	4	1,2,4,5	3	1,7	1	1	1
P19	3	2	4	2,5	1	1,4	1
P20	1	1,2	2	2,3,6	3	1,3	2

TABLE 10. Change record.

Version 1	SLR protocol
Version 1.1	Modification in inclusion and exclusion criteria, data extraction form and selection process

E. VALIDATION OF THE PROTOCOL

Table 10 shows the change record, i.e., the modifications incorporated in the review protocol, after the validation by the two expert reviewers.

F. SCHEDULE

We have started our work in September 2018 and completed the whole process in December 2018. Table 11 presents a detailed schedule of the review.

IV. FINDINGS AND DISCUSSIONS

The final stage of SLR is the detailed explanation, i.e., systematically analyzed the results of the research questions. With the help of this analysis, the researchers can incorporate the existing research efforts, and drive a research agenda for future development.

TABLE 11. Schedule.

Week(s)	Activity	Deliverable
1-2	Development of SLR Protocol, Given to Experts	Expert Review
3	Revision of SLR Protocol	Final SLR Protocol
4-6	Search for the publications	Publication list
7	Test/Re-test Process	NA
8-9	Selection Process	List of Selected studies
10-11	Data Extraction	Data Extraction Tables
12	Analyzing Data	NA
13-16	Reporting Results	Finish Report

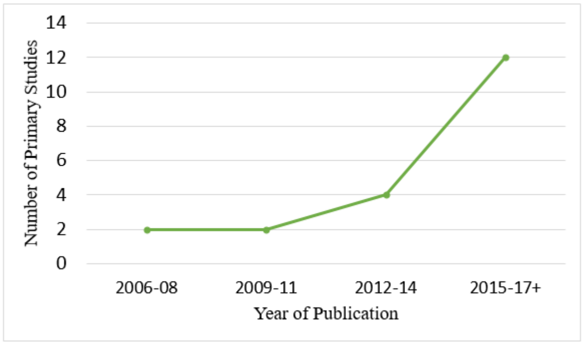

FIGURE 6. Distribution of studies over years.

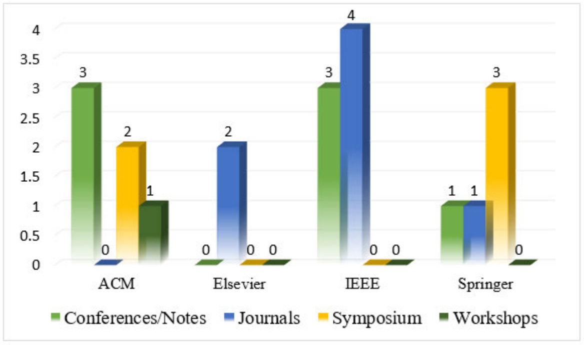

FIGURE 7. Research and publication trend of studies.

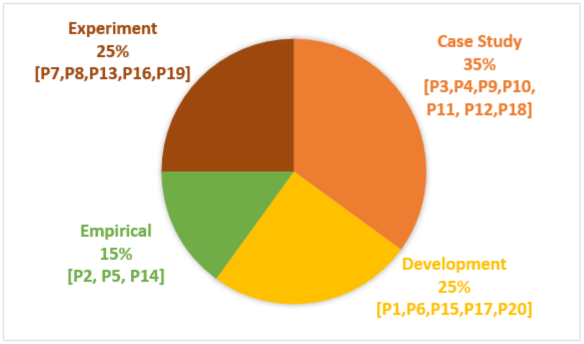

FIGURE 8. Distribution of studies according to applied research methodology.

A. RESEARCH TRENDS (RQ 1.1)

This subsection discusses the statistics of nominated studies, i.e., the source and distribution of publications. The work in GA based TCP techniques started in the late 2000s when Walcott *et al.* [13] in 2006 experimented with GA. They recorded a significant impact of GA usage in TCP. Li *et al.* [16], compared GA with other heuristics and showed GA helps in guiding the fitness function to add the next test case in prioritization order while taking into account the entire orderings. Henceforth, the researchers started working in this domain and found promising results. Wang *et al.* [7], [8], applied GA on industrial applications like software product lines testing and highly configurable systems testing e.g., Cisco Video Conferencing software. Fig. 6 shows a positive increment in the distribution of research work from July 2006 until the exploration of nominated studies for the review, i.e., August 2018.

Out of twenty selected studies, there are seven conference papers, seven journal articles, five symposium articles, and one workshop article, respectively. The most targeted sources of publication are IEEE Transactions, Springer Symposium on Search-Based Software Engineering, IEEE/ACM conferences, Elsevier Journal of System and Software and, so on. Fig. 7 summarizes the research and publication trend regarding the sources of publications and distribution of papers among publishers.

We categorize the papers according to applied research methodology, including the experiment based studies, the development based studies, empirical studies, and case studies. Fig. 8 shows that the majority of the work done on case studies (35%). The excellent performance of GA in industry applications related to TCP motivates researchers to explore this domain further. The next-highest proportion is the development based studies (25%) and the experimental studies (25%). The lowest (15%) are empirical studies that compare the performance of several algorithms with GA. Most of the studies applied default parameter settings, and only one research [29] tells about the effect of parameter settings of GA. Hence, there is a need for more empirical studies on parameter settings.

B. TCP CLASSIFICATION (RQ 1.2)

This sub-section provides the details of the TCP techniques used in the primary studies which are categorized as: code coverage, fault coverage, requirement coverage, configuration coverage, and a combination of these techniques. Fig. 9 represents the percentage of studies for the classification of TCP techniques.

Code Coverage: The most frequently used TCP techniques are code coverage based (55%) that order the test cases based on maximum coverage of structural elements, i.e., statements/blocks/methods/decisions/du-pairs. The maximal code coverage increases the chance of detecting almost all faults in the system. The different structural elements may produce different results, e.g., fine grain block coverage outperformed coarse grain method coverage [13]. A combination of these profiles might produce good results

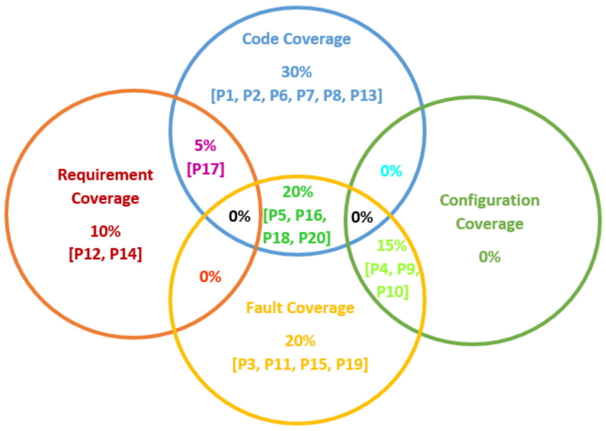

FIGURE 9. TCP classification.

but also selects a more percentage of tests to reveal more faults [38]. Li *et al.* [16], contradicted that the granularity did not affect the test case ordering, and one can opt any statement/block/decision coverage method. They did not consider the fault information. Bian *et al.* [21], showed that changed code coverage information has a similar impact on test case orderings as that of covering all statements of the code.

Fault Coverage: The next popular technique is fault coverage based (55%) methods that sort the test cases using historical information of faults. The more faults a test case covers, the more is its priority. In other words, the test case is more effective than other test cases if it has covered more faults in the previous version, comparatively. This method leads to the advent of popular performance metric Average Percentage of Fault Detection, i.e., APFD [2]. Khanna *et al.* [39], used historical information like fault severity (how much damage a fault can do to the software) and fault detection capability to produce more efficient sorting of test cases.

Configuration Coverage: Most of the industrial applications are configuration coverage based (15%) technique which performs the regression testing of configurable systems like Video Conferencing Software. Researchers prioritized test cases using different properties of the configuration systems, e.g., configuration coverage, status coverage, API coverage [28], [40], and efficient resource allocation [7].

Requirement Coverage: The least popular is requirement coverage based (15%) techniques that utilize the requirement specification information for prioritizing the test cases, e.g., coverage effectiveness, the customer assigned priority, and developer implementation complexity. There are only two primary studies based on requirement coverage. One is the empirical analysis of the role of GA operators for efficient TCP [29] and other traced requirements for cyber-physical software regression testing [41].

Others: These TCP techniques order the test cases by using two or more of the above coverage criteria (40%) or using some other criteria like feature pairwise coverage [8], [42]. Some of the researchers utilized the benefits of both fault and code coverage criteria for prioritization [15], [43], [44].

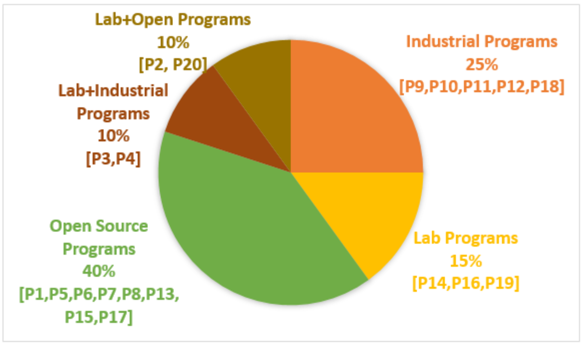

FIGURE 10. Distribution of studies according to dataset type.

Only one study combined the code and requirement coverage for identifying fault-prone areas and ordering the test cases accordingly [45].

C. TYPE AND GRANULARITY OF DATASET (RQ 2.1)

This sub-section outlines the type of dataset used, i.e., whether they are laboratory programs, industrial applications, and open-source projects. We also discuss the impact of the size of programs and the size of the prioritized test suite.

1) DATASET TYPE

Datasets are required to accomplish the controlled research on testing techniques. These may consist of test cases, source code, requirements, historical data, and fault information which depends upon the type of research work to experiment. Fig. 10 shows the distribution of studies according to the datasets used by various researchers in TCP. The open-source projects ($34 + 14 = 48\%$) have the highest proportion of datasets, e.g., JDepend (Java tool for quality metric) and Gradebook (a program for maintaining grades of course) [13], JTidy (HTML syntax checker) and NanoXML (XML parser) applications [38], open-source Java applications [45], Guava Java package [21], and most popular among these is Software Infrastructure Repository (SIR) [3], [11], [15], [16], [44], [46] which contains varying size and varying complexity programs. The next higher proportion is that of the real industrial datasets ($24 + 14 = 38\%$), i.e., Cisco Video Conferencing Software [7], [8], [28], [40], Drupal framework [42], and cyber-physical software [41], [47]. Out of 38% industrial studies, 14% studies investigated the scalability of the algorithms by changing the complexity of their own artificially produced laboratory/lab problems. However, due to confidentiality issues, industrial datasets cannot be reused. The remaining studies (14%) have used laboratory/lab programs (controlled experimentation with their private programs) for experimental purposes [29], [39], [43]. The datasets should be publicly available for the new researchers so that they can reproduce the research.

2) THE IMPACT OF GRANULARITY OF DATASET AND TEST SUITE

The test suite size and dataset size have a significant impact on experimental work. We have divided these sizes into

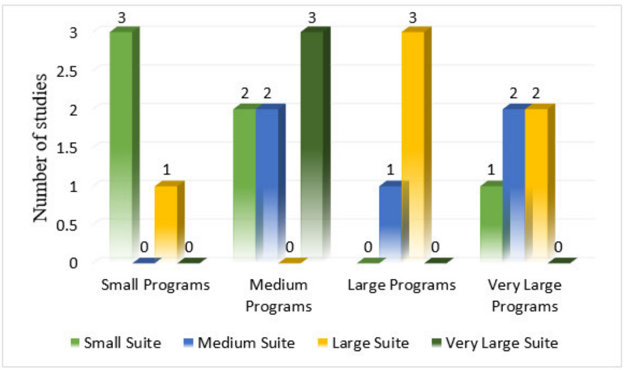

FIGURE 11. Relation between dataset size and test suite size.

four categories, i.e., small, medium, large, and very large size (see Table 8). Li *et al.* [16], found that a large size test suite affects the complexity due to an increase in the search-space time. They also noticed the indirect effect of program size on orderings because it increases the difficulty in fitness function calculations. Wang *et al.* [8], observed the same when they increased the number of features, the mean fitness value decreased, and when they increased the number of test resources, the mean fitness value increased. The performance of the algorithms [47] and execution time [15] increases with the increment in the size of the test suite, i.e., the algorithms are scalable to large and complex software. Marchetto *et al.* [45], found that test suite composition and fault distribution can also affect the results, e.g., test case redundancy can improve the results. Fig. 11 presents the distribution of studies according to the interrelation between the dataset size and the test suite size. Generally, the test suite size is proportionate to program size. Nucci *et al.* [15], noticed that the performance of GA is less than an additional greedy algorithm when a very large program has a small test suite. On the other hand, Masri and El-Ghali [38], used different datasets and GA settings, hence, they obtained contradictory results. It is suggested to check the robustness and scalability of the algorithm by applying it to varying sizes of programs and test suite.

D. TOOLS USED FOR IMPLEMENTATION (RQ 2.2)

This sub-section describes the tools used in nominated studies. We require tools for quick implementation and analysis of the work. Out of twenty studies, seven studies did not specify any tool information, and six studies used jMetal [7], [28], [40], [42]. jMetal is a Java-based framework for the implementation of meta-heuristic algorithms. Some researchers have integrated jMetal with other tools like Traceeclipse (for automatic recovery of traceability links) [45], and Matlab Simulink [41], etc., for both algorithmic and testing data implementation. Two studies [11], [46] used C language for algorithm implementation. Two studies [15], [44] used the Unix Diff tool, GNU C compiler, and gcov tools for tracking and collecting dynamic coverage information of inserted/deleted lines from the base version. Emma [13] and Cantataa++ [16] are other tools to collect code coverage information. Khanna *et al.* [39], used the Selenium framework for the automatic test case generation and other testing operations. Only one study used jTester for fault seeding purposes [13]. Table 12 presents a summary of the tools name and their requirement for different purposes. Because there is no standard tool for the collection and processing of testing information, one can use any tool which is appropriate for their research work.

TABLE 12. Tools used for implementation.

Tools Name	Requirement of the tool
jMetal, MATLAB Simulink, C, C++	Algorithmic implementation
TracEclipse, Emma, Cantataa++, Unix diff, GNU C Compiler and gcov	Code coverage Information
Selenium, jTester	Test case generation and fault seeding

E. GENETIC ALGORITHMS (RQ 3.1)

This sub-section briefs about the type of GA used by nominated studies for prioritizing the test cases. Fig. 12 presents the distribution of different types of GA with that of primary studies. Thirty percent of the studies used simple GA for ordering the test cases and, twenty-five percent of studies used modified/improved/enhanced GA in their experimentation/development of TCP. Walcott *et al.* [13] used the addition/deletion operators to extend the new generations beyond the initial generation. Pradhan *et al.* [28], found that the incorporation of clustering in GA can enhance the performance of the algorithm by producing steady or stable optimal solutions with the least number of fitness evaluations. NSGA-II used by various researchers (30%) for multi-objective prioritization, e.g., the combination of configuration and fault coverage [40]; requirement and execution cost [41]; fault and code coverage [42], [44]; code and requirement coverage [45]; code and execution cost [21]. Fifteen percent of the studies used two or more GAs at a time, e.g., Wang *et al.* [7] applied RWGA, NSGA-II, MoCell, and PAES to resource-aware TCP. They found different algorithms suitable for different objectives, but RWGA performed well when they combined all the objectives. Other researchers used NSGA-II and WBGA, [39] for web application testing, and, RWGA and WBGA, for highly configurable software [47].

F. PARAMETERS SETTING (RQ 3.2)

To map GA on TCP problem, one needs to consider several important factors, i.e., the nature of algorithm (the number of executions of the algorithm), representation of the problem, and type of operators required, the population and the generation size. This sub-section discusses these factors in detail.

1) STOCHASTIC NATURE OF GA

The evolutionary algorithms are stochastic. Therefore, the prioritization algorithm needs to be executed several times

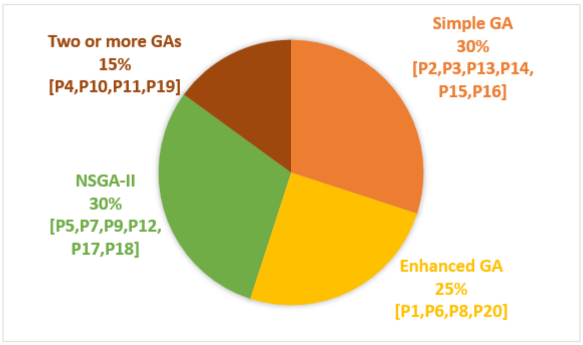

FIGURE 12. GAs used in TCP.

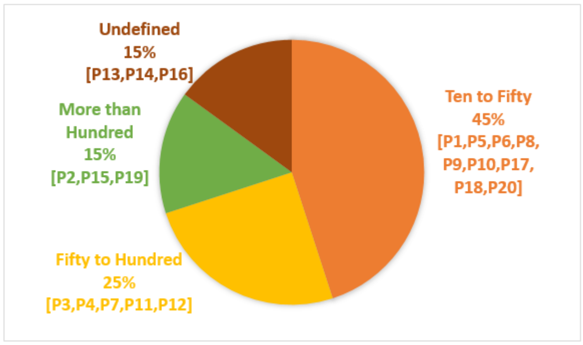

FIGURE 13. Distribution of studies according to number of executions of algorithm.

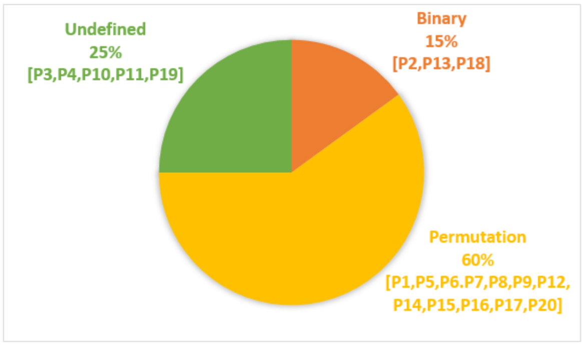

FIGURE 14. Type of encoding scheme used by studies.

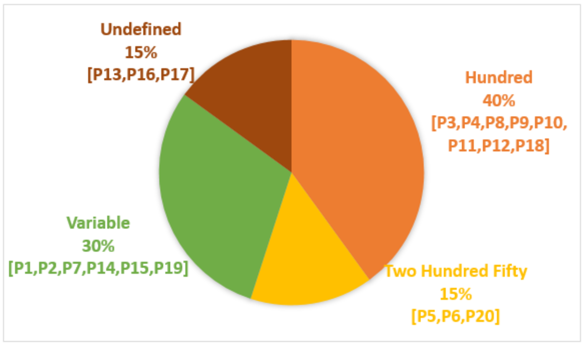

FIGURE 15. Distribution of studies according to population size.

for reliable results. Fig. 13 presents the distribution of the number of executions, i.e., how many times the algorithm has been repeated by the studies to get results. The majority of studies (45%) executed the algorithm between 10 to 50 times and, twenty-five percent run between 50 to 100 times. Industrial applications, i.e., large programs executed for more than 100 times (15%). The rest fifteen percent of studies did not mention how many times they implemented the algorithm. The range of the number of executions should be decided appropriately for reliable results.

2) POPULATION REPRESENTATION, POPULATION SIZE, AND GENERATION SIZE

The first step of the genetic algorithm to solve any problem is the representation of the population. Usually, it is represented in binary, decimal, integer, and character form. Fig. 14 shows that the most commonly used encoding scheme is permutation encoding (57%). In this, we do the numbering of test cases for population representation, i.e., integer form. 29% of studies did not define the representation form and, a few studies (14%) have used binary representation according to their experimental requirements [38], [42].

The population and generation size also influence the results. Different studies have used various sizes of population and generations. Most of the studies (40%) have set the population size equal to a hundred. Some of the researchers (30%) have used different values of population size to see the effect on the results and, 15% of studies used 250 as population size. The rest of the studies did not define the population size (see Fig. 15). It should not be too large and not too small because it may lead to premature convergence.

Generations decide the number of times the whole process of GA repeats, i.e., the stopping criteria. Twenty percent of studies used a generation size of fifty. Only 5% of studies used the generation size of a hundred and five hundred and, very large programs (15%) set it to a thousand. Thirty-five percent of studies showed the effect of the generation size by varying its value (see Fig. 16). Changes in generations and populations have a significant impact on time overheads. The fitness function evaluation is proportionate to the product of generation size and population size. A large generation size increased the crossover operations, which might give a chance to weak test cases, and ultimately increased the performance of the solution. It took less execution time as well [13], but if the population size is too small, then it may reverse the effect [11].

3) SELECTION OF OPERATORS

To reach an optimal solution, GA operators play an important role. They help in deciding where to move next to find the solution, i.e., the next generation. Three operators are there in GA, i.e., selection scheme, crossover operator, and mutation operator. The selection scheme selects individuals from the

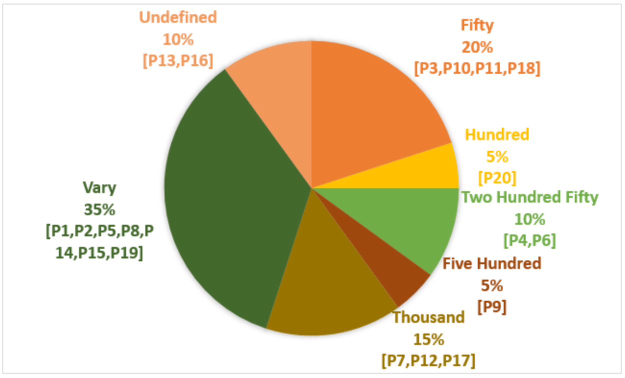

FIGURE 16. Distribution of studies according to generation size.

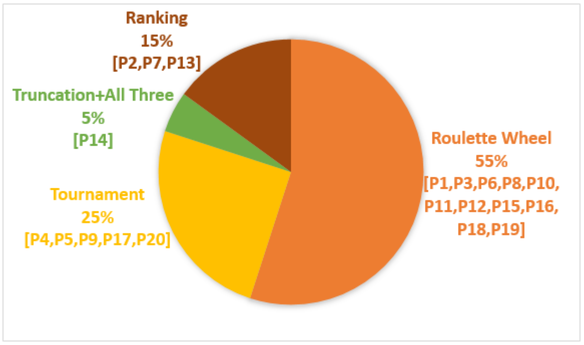

FIGURE 17. Type of selection scheme used by studies.

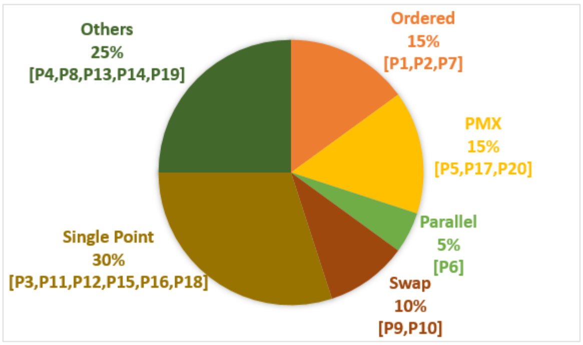

FIGURE 18. Type of crossover used by studies.

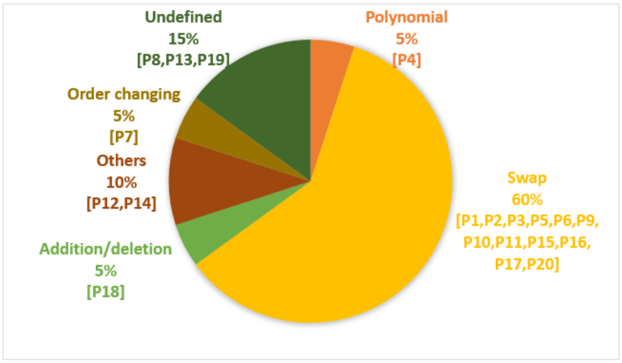

FIGURE 19. Type of mutation used by studies.

parent population to generate offspring for the new generation by using crossover operators and mutation operators. These selections can be roulette wheel, tournament, truncation, random, and rank-based. Most popular among these is roulette wheel selection (50%) which selects the individual based on their cumulative probability of fitness. The second most popular is tournament selection (29%) which runs several tournaments among the randomly selected individuals and use the winners for crossover. Rank-based selection (17%) ranks the individuals based on their fitness values and then selects the individuals of high rank for crossover. The least popular is truncation (4%), which discards some low fitness individuals of the parent population (see Fig. 17). One study [29] empirically analyzed the different selection pressures for requirement coverage based TCP. If the local optima problem is there with the low-intensity selection, then the use of the execution time and generation size as termination criteria may help in overcoming the problem. An increase in selection intensity improves the test case orderings.

After selection, the next step of GA is the crossover of two selected parents to generate the new offspring. The crossover operator produces new offspring by exchanging the genes of the parents. The most commonly used crossover operators are single-point crossover (32% of studies have used), two-point crossover, and uniform crossover. Besides the random exchange of genes, there are special crossover operators for the permutation encoding scheme. These are ordered crossover (16%), partially matched crossover (PMX) (16%) and swap crossover (10%). Twenty-one percent of studies used other types of crossovers, e.g., simulated binary [29] (see Fig. 18). Some researchers modified the crossover operators to enhance the performance of the solution, e.g., Li *et al.* [46], developed three parallel crossover schemes which work better than a series crossover. However, the proposed parallel general crossover had unstable execution time due to the complexity of programs. Yuan *et al.* [3], applied epistasis theory to one point (E-SC) and two-point crossover operator (E-Ord) and compared it with one point, Ordered and PMX crossover. E-Ord took more time to reach a stable state than Ordered crossover. However, it always gave higher fitness value. It is significantly better than the PMX crossover because it requires less number of iterations. The promising results suggest that more experiments need to be done to enhance the operator's efficiency.

A mutation operator adds genetic diversity to the new generation. The mutation operator alters the genes of the individuals to create better individuals. The most popularly used mutation operator is swap mutation (57%). There are different types of mutation operators such as Addition/Deletion (5%), Order changing (5%), polynomial (5%), and others (14%). Fourteen percent of the studies did not describe the type of mutation used (see Fig. 19).

The crossover rate and mutation rate plays a crucial role in deciding the direction of searching for the solution. Forty percent of the studies used a crossover rate of 0.9, and

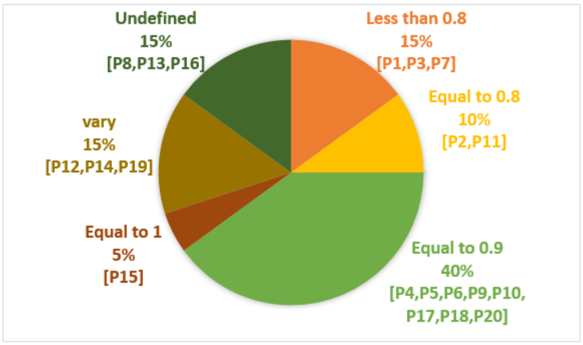

FIGURE 20. Distribution of studies according to different crossover rates used.

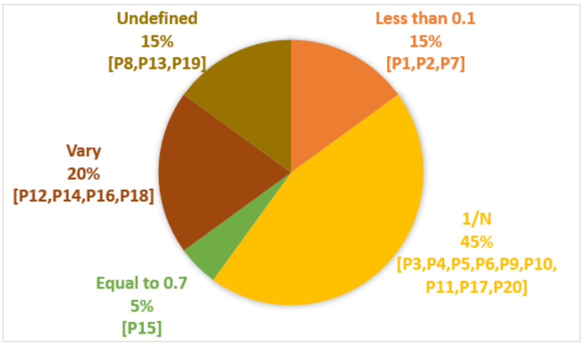

FIGURE 21. Distribution of studies according to different mutation rates used.

twenty-five percent of studies used rate less than equal to 0.8. Fifteen percent of the studies did not describe the crossover rate of their work. Last but not least, 15% of studies used variable values. Fig. 20 shows the distribution of crossover rate among selected studies.

Similarly, forty-five percent of studies used the mutation rate of 1/N (where N = number of test cases). Twenty percent experimented with different mutation rates and crossover rates to check the valid value for their work. For example, Arrieta *et al.* [41], found that the best mutation rate was 2/N, and, the corresponding crossover rate was 0.2 for their work. Though this crossover rate did not affect much but statistically took less running time than other crossover rates. Most of the studies used default parameter settings and, only a few studies empirically analyzed the operators and the effect of their rates. The selection of operators and their probabilistic rates require special attention, i.e., the value should not be too low or too high. Fig. 21 shows the different mutation rates used by nominated studies.

G. FITNESS FUNCTION DESIGN (RQ 3.3)

Fitness function finds out how close the given solution is from the optimal solution. For the single-objective approach, the fitness function is the objective function of the problem, which either maximizes or minimizes to get optimal solutions. On the other hand, the multi-objective approach has multiple objective functions for each goal, i.e., either maximize or minimize individually to give the trade-off solutions. The multi-objective approach can behave like a single-objective approach if we combine the objective functions into a single fitness function by assigning weights to each objective according to their goal [8].

Fitness function design is the crucial stage in optimizing any problem, so one has to design it carefully because the final result depends on it. If not defined efficiently, then it may lead to several issues. For example, improper design of fitness function may produce the wrong solution or may stick the solution at local optima. In other words, the fitness function can evaluate lower fitted solutions better even if the more fitted solution exists. This problem mainly arises due to the small size of the population. There is no particular rule to design the fitness function as it depends on the type of problem in hand. TCP problem prioritizes the test cases according to some testing coverage criteria and corresponding to that, we have coverage/performance metrics and/or the cost-effectiveness measures, which may lead to the formulation of fitness functions. In other words, each nominated study designed its fitness function with the help of the coverage/performance metrics and/or cost-effectiveness measures by considering the defined coverage criteria. For the list of fitness functions used in nominated studies refer to Table 13.

From the analysis of selected studies, it is evident that the single objective optimization TCP problems used coverage/performance metrics, e.g., APFD [11] for fault coverage, CE [29] for requirement coverage and APSC [3] for statement coverage as a fitness function, respectively. The multi-objective TCP provides a trade-off among the various objectives, so the studies [21], [28], [39]–[42], [44]–[46] designed multiple objective functions for different solutions utilizing the coverage/performance metrics and/or cost-effectiveness measures. For example, Li *et al.* [46], found the trade-off solutions by maximizing the code coverage using the APSC metric and minimizing the execution cost simultaneously. Pradhan *et al.* [28], [40], maximized the objective functions using the metrics of configuration coverage, test API coverage, status coverage, and FDC, respectively. Some of the studies [7], [8], [13], [15], [43], [47] treated the multi-objective problem as a single objective by using the weighted sum of the performance metrics and/or cost-effectiveness measures to calculate the fitness. For example, Wang *et al.* [7], used the normalized weighted sum of total time, prioritization density, test resource usage, and FDC. Whereas, Nucci *et al.* [15], used the hypervolume indicator to condense multiple objectives to a single objective.

H. PERFORMANCE METRICS (RQ 4.1)

This sub-section explains the type of performance metrics used to evaluate the effectiveness of GA based TCP approaches. Fig. 22 presents the distribution of performance

TABLE 13. Fitness function designs used in the studies.

ID	Coverage Criteria	Number of Fitness Functions	Fitness Function Design [1]
P1	Code coverage	Single	The fitness function was calculated by adding two functions (i) Primary Fitness which measures the code coverage of entire test suite (ii) Secondary Fitness which measures the incremental code coverage giving priority to those test cases which have greater coverage.
P2	Code coverage	Single	Banker's formula with selection pressure=2 was used as fitness function, i.e., 2(Pos-1)/(Nind-1) where Pos represents the individual and the Nind is the population size.
P3	Fault coverage	Single	The fitness function is calculated as sum of weighted cost-effectiveness measures (after normalization), i.e., Overall Execution Cost, Prioritized Extent, Feature Pairwise Coverage and Fault Detection Capability.
P4	Fault and configuration coverage	Single	The normalized weighted sum of cost-effectiveness measures (Total Time, Prioritizing Density, Test Resource Usage, and Fault Detection Capability) was used as fitness function.
P5	Code and fault coverage	Multiple	Three Objective functions were designed for evaluation (i) Statement Coverage (ii) Difference of the Statement Coverage (iii) Fault History Coverage. Each of these was calculated using the same formula as that of APFDc where c=execution cost of the test case.
P6	Code coverage	Multiple	The fitness function is designed with the help of performance metrics Average Percentage of Statement Coverage and Effective Execution Time
P7	Code coverage	Multiple	The fitness was calculated using the Effective Execution Time, Statement Coverage and Condition Coverage using the same formula as that of APFD calculation
P8	Code coverage	Single	APSC metric had been used as fitness function
P9, P10	Fault and configuration coverage	Multiple	Fitness was calculated using four objective functions which were defined as maximizing (i) Configuration Coverage (ii) Test API Coverage (iii) Status Coverage (iv) Fault Detection Capability
P11	Fault coverage	Single	The fitness function was calculated as weighted sum of Fault Detection Capability and the Execution Time of the test case.
P12	Requirement Coverage	Multiple	The cost-effectiveness measures, i.e., requirement coverage, test case similarity, prioritization-aware similarity and test execution time were used for fitness calculation
P13	Code coverage	Single	The fitness was calculated as 1-%tests where %tests denoted the percentage of test cases which exercised the combination of program elements.
P14	Requirement coverage	Single	The Coverage Effectiveness Score was used for fitness function which is calculated by dividing the integral of Cumulative coverage of requirements for given time units. with Ideal cumulative coverage, i.e., immediate coverage of all the requirements.
P15	Fault coverage	Single	The fitness function used the formula of APFD calculation and also considered the fault severity of the fault in previous testing and the cost of the each test case of the test suite.
P16	Code and fault coverage	Single	Fitness function was calculated as the weighted sum of maximum coverage of conditions, multiple conditions and statements in each test case and this weight is further divided by order of the test case.
P17	Code and requirement coverage	Multiple	Area Under Curves of cumulative code coverage, cumulative requirement coverage and inverse cost were used as fitness function.
P18	Code, fault, configuration coverage	Multiple	Several functional objectives like coefficient of connectivity density, dissimilarity, pairwise coverage etc. and non-functional objectives like number of changes, number of faults and feature size were used to design fitness function.
P19	Fault coverage	Multiple	The severity of the fault, execution cost of test case and the APFDc of the test suite were used for fitness function calculation.
P20	Code and fault coverage	Single	The fitness function was designed for two to five criteria problem by taking hypervolume of execution cost, statement coverage, past fault coverage, branch coverage and function coverage.

[1]See corresponding references for mathematical formula of the metrics used in designing fitness function.

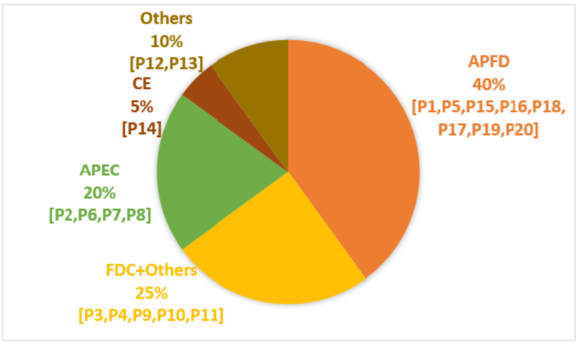

FIGURE 22. Distribution of studies according to performance metrics.

metrics of primary studies. Commonly used performance metrics for evaluation of TCP techniques are:

(1) *Average Percentage of Fault Detection:* "Average Percentage of Fault Detection (APFD) calculates the weighted average of the percentage of faults detected by the test cases of the test suite [2]." The original APFD metric does not incorporate the fault severity and test case cost so, Elbaum *et al.* [48], modified it and named APFDc. In other words, APFDc is the cost-aware version of APFD, which considers the fault severity and the execution cost of the test cases. APFD is the most popular metric used by 35% of studies for evaluation purposes.

(2) *Fault Detection Capability:* The next popular metric is Fault Detection Capability (FDC), which covers 25% of studies. It is different from the APFD metric in a way that it finds the rate of success of the test suite, i.e., if a test case detects a fault, then it succeeds else fails. It uses the historical information of the execution of test cases. It is the proportion of the number of times a test case found a fault to the number of times it was executed. For example, if a test case performed for six times, out of 20 executions, then its FDC is 0.3 [40].

TABLE 14. Studies considered execution cost as performance metric.

Execution Cost	Studies
Yes	P1, P3, P4, P6, P7, P11, P14, P15, P19, P20
No	P2, P5, P8, P9, P10, P12, P13, P16, P17, P18

(3) *Average Percentage of Element Coverage:* An Average Percentage of Element Coverage (APEC) is the percentage at which the ordered test suite covers the elements, i.e., statements/blocks/decisions. This metric is similar to the APFD. However, the calculation is done on the element coverage instead of the fault coverage [16]. The proportion of papers that used APEC as a metric is 25%.

(4) *Coverage Effectiveness:* Coverage Effectiveness (CE) is the proportion of the prioritized test suite to the ideal test suite which covers all the requirements. There is only one paper [29] which utilized CE for requirement coverage.

(5) *Others:* Various researchers also used some other metrics, e.g., feature pairwise coverage, prioritizing extent [8], test resource usage [7], the percentage of test order diversity [45]. Researchers have also used quality indicators for performance measurement, e.g., Epsilon, Hypervolume, Inverted Generational Distance (IGD) [44]. In addition to these metrics, 50% of studies have included the execution cost of test cases as another performance metric for prioritizing the test cases, which is summarized in Table 14.

I. VALIDATION OF ALGORITHM (RQ 4.2)

This sub-section summarizes how the researchers of nominated studies validated the results. To prove the effectiveness of the proposed algorithm, one needs to compare its performance with other like algorithms or check its efficiency by varying its operators or different parameter settings. Fig. 23 shows the distribution of studies according to their validation criteria. In other words, whether the researchers have compared GA with traditional algorithms (e.g. greedy algorithm), different search-based algorithms (e.g. Two Archive Evolutionary Algorithm (TAEA)), or with already existing work. A large proportion (75%) of studies have compared GA with other algorithms. Out of which 40% of studies also checked the efficiency of the algorithm. The rest 25% of studies only examined the efficiency of the algorithm by varying its operators, population size, and generation size. Table 15 shows the comparison of GA with other algorithms and their performance. The table concluded that GAs outperformed random searches in all case studies and all objectives. Except in some cases, they also outperformed traditional algorithms and other search algorithms.

J. STATISTICAL SIGNIFICANCE (RQ 4.3)

We observed that 60% of the studies proved the better performance of the algorithms by statistical analysis. The majority of them used Vargha and Delaney test and Mann-Whitney U test [21], [28], [40], [41]. In addition to these, some also used Spearman's correlation test [7], [8], [47]. Only one study [15] used Welch's t-test, Wilcoxon t-test along with Vargha and

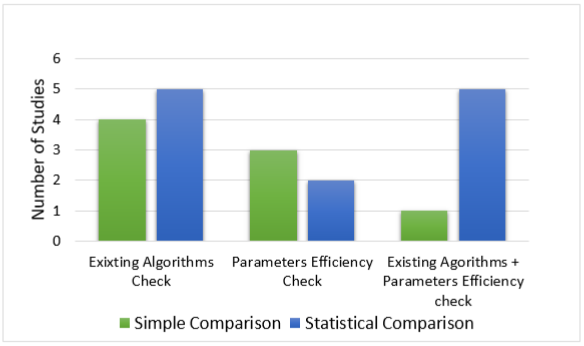

FIGURE 23. Distribution of studies according to validation criteria.

Delaney test for the many-criteria problem. They also used a two-way permutation test [45] to check whether the difference between the execution time of two algorithms significantly interacts with the test suite size. Epitropakis *et al.* [44], used Vargha and Delaney test, and Wilcoxon signed rank test with Bonferroni adjustment for type 1 error.

Researchers also used ANOVA and LSD [16]. Conrad *et al.* [29], used the tree model to analyze the effect of the selection operator on the performance of the algorithm. Wang *et al.* [7], performed the Kruskal-Wallis test along with Bonferroni correction to study the statistical difference of running time of search algorithms.

K. MAJOR FINDINGS, PURPOSE AND LIMITATIONS OF STUDIES (RQ 5.1)

This sub-section outlines the strength and weaknesses exhibited by the GA based TCP techniques as recorded by researchers. We have extracted the major findings and limitations and factors to be resolved in the future of nominated studies and discussed in appropriate sections. Table 6 presents the purpose of studies, i.e., why and where GA has been applied.

We have taken the limitations of studies from the threats to the validity section of selected studies. Different researchers found some common weaknesses, e.g., use of default parameter settings, mutated faults, use of specific tools, coverage information used, type and number of case studies used to validate the results. These limitations may lead to biased results if we apply the algorithm in different conditions than the defined one. In other words, the results are not in a generalized form. Table 16 lists common threats and their corresponding studies.

L. RESEARCH DISCUSSIONS AND SUGGESTIONS (RQ 5.2)

We have conducted this review to indicate significant directions on how to improve the quality of GA based TCP. From review observations, it is evident that GAs have great potential in solving TCP problems, and the area is still open for improvements. Future researchers can opt this field as it has proved its performance for single-objective and multi-objective real-world industrial applications. To apply

GA in TCP, the researcher should be aware of the appropriate operators and the optimal values of the parameters. If the researcher is successful in using GA in TCP with proper parameter settings, then the regression testing can be performed efficiently and effectively and within the stipulated time. Based on the review, here, we give some suggestions regarding the use of GA in TCP, which may help future researchers for proper utilization of the benefits of the GA.

TABLE 15. **Comparison table.**

ID	Type of GA used	Compared With	Performance Metrics	Results
Single-Objective GA				
P2	GA	Greedy, Additional Greedy, 2-optimal, Hill Climbing	APEC	GA performed well but Greedy was much effective due to multimodal landscape of TCP (code coverage).
P11	WBGA, RWGA	Random Search (RS), Alternate Variable Machine (AVM), Greedy algorithm	Test Case Execution Time, FDC	Local search methods were better than global search methods.
P15	GA	RS, optimal search, and techniques developed by different researchers	APFDc	With the help of mutation operator GA give chances to those test cases which have lower historical records.
Multi-Objective GA				
P3	GA	RS, (1+1)Evolutionary Algorithm, AVM	Overall Execution Cost, Prioritized Extent, Feature Pairwise Coverage, FDC	GA significantly outperformed AVM but not than (1+1)EA.
P4	RWGA, NSGA-II, MoCell, SPEA2, PAES	CellDE, SMPSO	Total time (TT), Prioritization Density (PD), Test Resource Usage (TRU), FDC	Different algorithm performs well for different objectives, e.g., for TT: RWGA, TRU: MoCell, PD: NSGA-II, SPEA2, FDC: CellDE and for all objectives: RWGA.
P5	NSGA-II	RS, TAEA, and two hybrid Multi-objective Evolutionary Algorithms (MOEA) with Additional Greedy seeding	APFDc, Epsilon, Hypervolume (HV), Inverted Generational Distance	TAEA better than NSGA-II in effect size comparison but both are better than additional greedy.
P9	NSGA-II	RS, Greedy algorithm, and techniques developed by different researchers.	Modified FDC, Status coverage (SC), Configuration coverage (CC), API coverage (APIC)	The proposed method outperformed for the four defined time budgets (25%, 50%, 75%, 100%).
P10	CBGA-ES	RS, NSGA-II, MoCell, SPEA2, PAES	HV, FDC, SC, APIC, CC	CBGA-ES surpassed other techniques for all objectives.
P12	NSGA-II	RS, NSGA-III, SPEA2, MOEA/D, PESA-II	HV, Requirement Coverage, Total Execution Time, Test Case Similarity, Prioritization Aware Test Case Similarity	NSGA-III gave worst performance for HV. Some algorithms performed better than NSGA-II for different case studies and objectives but overall NSGA-II is the winner.
P17	NSGA-II	RS, Code-Cov, Add-CodeCov	APFD, Percentage of Test Order Diversity	The comparison was done on the basis of the Pareto Metrics, robustness, effectiveness, sensitivity, running time, and co-factors and all results favored the proposed approach.
P18	NSGA-II	RS, techniques developed by other researchers	APFD	Multi-objective TCP outperformed for both functional and non-functional objectives and their combination. NSGA-II using functional coverage was slightly worse than non-functional coverage.
P19	NSGA-II, WBGA	RS, Greedy, Additional Greedy, 2-optimal, improved 2-optimal	Execution Cost, Fault Severity, APFD	NSGA-II outperformed in all cases. WBGA performs equivalent to NSGA -II in some cases but in others it showed less performance.
P20	GA using Hyper-volume indicator	GA, Additional Greedy, NSGA-II, MOEA/D-DE, GDE3	APSCc, APFDc, HV	The proposed algorithm gave mixed results for two, three, four and five criteria problem in terms of running time and HV but it is better in case of very large programs.

(i) *Need of more empirical studies*
Most of the research has been conducted for development/experimental purposes, but less emphasis is given on empirical studies. It is suggested to compare GA with other state-of-the-art techniques to prove its effectiveness.

(ii) *Develop more requirement based TCP methods*
It is observed that GA has been applied for structural coverage, fault coverage, and configuration coverage of test cases. However, the maximum number of studies are based on code coverage. Recent studies [49], [50] have proved that code coverage is not sufficient for covering all the faults in software. Therefore, we can conclude

TABLE 16. Limitations of the studies.

S. No.	Limitations	Studies
1	Use of default parameter settings.	P3, P4, P9, P10, P11, P18, P20
2	Use of mutated faults instead of real faults.	P1, P11, P17, P19
3	Use of same stopping criteria.	P3, P4, P9, P10, P11, P12
4	Use of specific tools and programs under test.	P2, P5, P8
5	Use of specific type of coverage criteria so may produce baised results if used in other conditions.	P2, P8, P14
6	Experiment with only one or two case studies and very small size programs.	P9, P11, P14, P15, P16, P18
7	In empirical analysis, the researchers have used different parameter settings for different algorithms which may lead to different performance of the algorithms.	P12
8	Use of only one performance metric.	P1, P8, P20

that the prioritization method should consider more than two coverage criteria for better performance. It is also observed that only two studies have worked upon requirement coverage. Future researchers can focus on these areas for further research.

(iii) *Use of Public Datasets*
Industrial applications show that the usage of GA is fruitful for the TCP. However, they do not publicize their datasets. Because of this, their results are not reproducible, and the researchers cannot validate the findings with the prior works. It is suggested to use public datasets that can reproduce results if required. For example, the SIR repository is popular among the researchers and is available in the public domain.

(iv) *Generalization of results*
It is observed that the researchers validate the algorithm on one or two case studies only, and the algorithm responds differently for different case studies. It is suggested to conduct the experiment on more than two case studies and try to generalize the results, i.e., the algorithm should apply to other datasets also.

(v) *Scalability of the algorithm*
Some researchers have checked the efficiency of the algorithm by varying the size of programs and test suite, i.e., their technique is scalable to programs of different complexity. For example, one researcher found a decrease in performance when he used a small test suite of a very large program. Future researchers can empirically analyze this factor by varying the size of the test suite as well as programs under test.

(vi) *Use of tools*
Different researchers have used different tools for their implementation. There are no standard tools available, so there may be a chance of bias if different tools are used. It is suggested to draw some benchmark guidelines for the use of tools.

(vii) *Genetic Algorithms*
The researchers have utilized the benefits of GAs by applying it to both single-objective and multi-objective problems. The performance of GA can be enhanced by applying parallelism and clustering. It is suggested to use simple GA first then go for further revisions or variants which is time savvy.

(viii) *Parameters and Operators Settings*
The researchers have observed that the parameter settings and operators' role have a significant impact on the performance of the algorithm. It is required to do more empirical analysis on the population representation, population size, generation size, operators, and their probabilistic rates to make the base guidelines. As GA is stochastic, so to reduce the randomness of the results, the algorithm must be repeated for some specific range, and the average result can be used for performance measurement. One can also use automated tools for optimal parameter and operator settings, e.g., irace package [51].

(ix) *Fitness function design*
Fitness functions should be adequately defined because improper design may lead to wrong results. It is observed that the fitness function in TCP was designed with the help of cost-effectiveness measures. So, the proper weight tuning and the parameter tuning need to be done as they directly influence the fitness function calculations.

(x) *Performance Metrics*
Most of the researchers have used a single performance metric, i.e., APFD. It is suggested to check the efficiency of the algorithms by applying other famous performance metrics like APEC, FDC, and CE. Researchers can use more than two metrics to prove the robustness of the algorithm.

(xi) *Validation and Statistical Analysis of algorithm*
Some researchers have compared GA with random search and other algorithms. To find out the best algorithm, one needs to compare it with different existing algorithms. It is a good practice if one examines one's algorithm with at least two other algorithms in addition to random search. The statistical analysis adds more flavor to validation because it tells an exact difference between the two algorithms. It is suggested to apply statistical tests for making comparisons.

V. THREATS TO VALIDITY

Internal validity deals with the use of repositories and the formation of the search string. Different authors have used different keywords for the same research subject. Hence, it is difficult to find relevant studies from various sources. The authors have tried to control this threat by performing the formal search through keywords in popular databases, succeeded by the snowballing procedure to select all the relevant studies. Decisions made during the drafting of the protocol also affect the results drawn from the SLR. In other words, a different researcher may end up selecting different studies. To overcome this risk, one author (Ph.D. supervisor) has ensured the relevant sampling of the results other than the principal author. This sampling process is also supplemented by implementing the test/retest process.

Construct validity is concerned with the construction of research questions and the data extraction process. This risk is minimized by adopting the PICOC criteria, and the quality assessment score for the data extraction process because the formation of research questions mainly rely upon the extracted data. Though the quality checklist is not precise enough to assess the quality of studies, it has the potential to differentiate between the better and weaker studies. The researchers' bias may also lie in answering the quality questions objectively. To overcome this threat, both the authors have scored the studies independently, and the average score (after discussion) is used for selecting the relevant studies. Another potential risk lies in the imprecise data synthesis and validation process. For this, we have developed a structured format and classified all the relevant studies into seven classes to reduce the possibility of inaccurate data extraction.

Conclusion validity lies in the correctness of the conclusions. The analysis is limited to only those studies which have mentioned the GA usage in TCP. The authors argue that this study is aimed to exploit and explore the research directions for GA application in TCP which can help researchers in the selection of parameters, operators, application areas, and overcoming the limitations of using GA in TCP.

External validity deals with the extent to which the results can be generalized. The authors have not included other approaches to generalize the results. To limit this threat, we have added the relevant studies from the popular sources which are concerned about the use of GA in TCP so that in-depth discussion on the topic and better future directions can be laid out.

VI. CONCLUSION

Regression testing is one of the vital processes of software quality assurance and maintenance. Test case prioritization has gained the interest of researchers for regression testing as it only rearranges the test cases instead of permanent removal. TCP using meta-heuristics algorithms, especially nature-inspired algorithms like GA, have been widely explored by various researchers. As the number of publications in GA based TCP is increasing, it is high time to summarize the existing research for new researchers. Therefore, we have performed SLR to provide a detailed analysis of TCP in conjunction with GAs by formally evaluating and interpreting the research done so far. This secondary study aimed at classifying and criticizing the current state of research, by collecting pieces of evidence of the use of GA in TCP, and finding out the areas which need to be explored or exploited in the future. We have used a well-defined SLR protocol to present the best available pieces of evidence and identified 20 primary studies that have used GA in TCP. We have formulated several research questions according to the goal of the SLR and tried to answer them by synchronizing the extracted data with the help of tabular and graphic representation. To ease the process, we have also classified the selected studies into seven classes, namely, research methodology, the prioritization method, type of GA used, dataset specification, test suite size, performance metrics, and validation criteria. Furthermore, the most critical aspects of GA have been discussed like the population representation, population size, and generation size, types of operators used, operators' probabilistic rate, and fitness function design, respectively. The impact of coverage criteria, granularity of dataset size and test suite size, and tools were critically analyzed. We have also examined the pros and cons of studies and provided some suggestions. In summary, this review presented insight into designing GA for TCP and put forth some ideas for future researchers to fully utilize the potentials of GA based TCP.

REFERENCES

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

[2] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, pp. 159–182, Feb. 2002.

[3] F. Yuan, Y. Bian, Z. Li, and R. Zhao, "Epistatic genetic algorithm for test case prioritization," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2015, pp. 109–124.

[4] M. Harman and B. F. Jones, "Search-based software engineering," *Inf. Softw. Tech.*, vol. 43, no. 14, pp. 833–839, 2001.

[5] A. Bajaj and O. P. Sangwan, "A survey on regression testing using nature-inspired approaches," in *Proc. 4th Int. Conf. Comput., Commun. Automat.*, 2018, pp. 1–5.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[7] S. Wang, S. Ali, T. Yue, Ø. Bakkeli, and M. Liaaen, "Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search," in *Proc. 38th Int. Conf. Softw. Eng. Companion*, 2016, pp. 182–191.

[8] S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan, and M. Liaaen, "Multi-objective test prioritization in software product line testing: An industrial case study," in *Proc. 18th Int. Softw. Product Line Conf.*, vol. 1, 2014, pp. 32–41.

[9] S. M. Thede, "An introduction to genetic algorithms," *J. Comput. Sci. Colleges*, vol. 20, no. 1, pp. 115–123, 2004.

[10] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM Syst. J.*, vol. 41, no. 1, pp. 4–12, 2002.

[11] Y. C. Huang, K. L. Peng, and C. Y. Huang, "A history-based cost-cognizant test case prioritization technique in regression testing," *J. Syst. Softw.*, vol. 85, no. 3, pp. 626–637, 2012.

[12] Y. Singh, A. Kaur, B. Suri, and S. Singhal, "Systematic literature review on regression test prioritization techniques," *Informatica*, vol. 36, no. 4, pp. 379–408, 2012.

[13] K. R. Walcott, M. L. Soffa, G. M. Kapfhammer, and R. S. Roos, "Timeaware test suite prioritization," in *Proc. Int. Symp. Softw. Test. Anal.*, 2006, pp. 1–12.

[14] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: An empirical study," in *Proc. Int. Conf. Softw. Maintenance*, 1999, pp. 179–188.

[15] D. Di Nucci, A. Panichella, A. Zaidman, and A. De Lucia, "A test case prioritization genetic algorithm guided by the hypervolume indicator," *IEEE Trans. Softw. Eng.*, to be published.

[16] Z. Li, M. Harman, and R. M. Hierons, "Search algorithms for regression test case prioritization," *IEEE Trans. Softw. Eng.*, vol. 33, no. 4, pp. 225–237, Apr. 2007.

[17] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[18] L. S. Ochi, D. S. Vianna, L. M. A. Drummond, and A. Victor, "A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet," *Future Gener. Comput. Syst.*, vol. 14, nos. 5–6, pp. 285–292, 1998.

[19] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.

[20] L. Briand, Y. Labiche, and K. Chen, "A multi-objective genetic algorithm to rank state-based test cases," in *Proc. Int. Symp. Search Based Softw. Eng.* Berlin, Germany: Springer, 2013, pp. 66–80.

[21] Y. Bian, S. Kirbas, M. Harman, Y. Jia, and Z. Li, "Regression test case prioritisation for Guava," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2015, pp. 221–227.

[22] D. E. Goldberg and R. Lingle, Jr., "Alleleslociand the traveling salesman problem," in *Proc. Int. Conf. Genetic Algorithms Appl.*, Hillsdale, NJ, USA, 1985, pp. 154–159.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[24] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Dept. Elect. Eng., ETH Zürich, Zürich, Switzerland, TIK-Rep. 103, 2001.

[25] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "MOCell: A cellular genetic algorithm for multiobjective optimization," *Int. J. Intell. Syst.*, vol. 24, no. 7, pp. 726–746, 2009.

[26] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, Jun. 2000.

[27] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: Region-based selection in evolutionary multi objective optimization," in *Proc. Genetic Evol. Comput. Conf.*, San Francisco, CA, USA, 2001, pp. 283–290.

[28] D. Pradhan, S. Wang, S. Ali, T. Yue, and M. Liaaen, "CBGA-ES: A cluster-based genetic algorithm with elitist selection for supporting multi-objective test optimization," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation*, Mar. 2017, pp. 367–378.

[29] A. P. Conrad, R. S. Roos, and G. M. Kapfhammer, "Empirically studying the role of selection operators duringsearch-based test suite prioritization," in *Proc. 12th Annu. Conf. Genetic Evol. Comput.*, 2010, pp. 1373–1380.

[30] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," *Inf. Softw. Technol.*, vol. 93, pp. 74–93, Jan. 2018.

[31] C. Catal and D. Mishra, "Test case prioritization: A systematic mapping study," *Softw. Qual. J.*, vol. 21, no. 3, pp. 445–478, 2013.

[32] C. Catal, "On the application of genetic algorithms for test case prioritization: A systematic literature review," in *Proc. 2nd Int. Workshop Evidential Assessment Softw. Technol.*, 2012, pp. 9–14.

[33] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ., Keele, U.K., Tech. Rep. EBSE 2007-001, 2007, vol. 5.

[34] J. Buckley, T. Mens, M. Zenger, A. Rashid, and G. Kniesel, "Towards a taxonomy of software change," *J. Softw. Maintenance Evol., Res. Pract.*, vol. 17, no. 5, pp. 309–332, 2005.

[35] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. ACM 18th Int. Conf. Eval. Assessment Softw. Eng.*, 2014, Art. no. 38.

[36] C. Wohlin, "Second-generation systematic literature studies using snowballing," in *Proc. ACM 20th Int. Conf. Eval. Assessment Softw. Eng.*, 2016, Art. no. 15.

[37] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *J. Syst. Softw.*, vol. 80, no. 4, pp. 571–583, 2007.

[38] W. Masri and M. El-Ghali, "Test case filtering and prioritization based on coverage of combinations of program elements," in *Proc. 7th Int. Workshop Dyn. Anal.*, 2009, pp. 29–34.

[39] M. Khanna, N. Chauhan, D. Sharma, A. Toofani, and A. Chaudhary, "Search for prioritized test cases in multi-objective environment during Web application testing," *Arabian J. Sci. Eng.*, vol. 43, no. 8, pp. 4179–4201, 2018.

[40] D. Pradhan, S. Wang, S. Ali, T. Yue, and M. Liaaen, "STIPI: Using search to prioritize test cases based on multi-objectives derived from industrial practice," in *Proc. IFIP Int. Conf. Testing Softw. Syst.*, 2016, pp. 172–190.

[41] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui, and L. Etxeberria, "Employing multi-objective search to enhance reactive test case generation and prioritization for testing industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1055–1066, Mar. 2018.

[42] J. A. Parejo, A. B. Sánchez, S. Segura, A. Ruiz-Cortés, R. E. Lopez-Herrejon, and A. Egyed, "Multi-objective test case prioritization in highly configurable systems: A case study," *J. Syst. Softw.*, vol. 122, pp. 287–310, Dec. 2016.

[43] A. A. Ahmed, M. Shaheen, and E. Kosba, "Software testing suite prioritization using multi-criteria fitness function," in *Proc. 22nd Int. Conf. Comput. Theory Appl.*, 2012, pp. 160–166.

[44] M. G. Epitropakis, S. Yoo, M. Harman, and E. K. Burke, "Empirical evaluation of Pareto efficient multi-objective regression test case prioritisation," in *Proc. Int. Symp. Softw. Test. Anal.*, 2015, pp. 234–245.

[45] A. Marchetto, M. M. Islam, W. Asghar, A. Susi, and G. Scanniello, "A multi-objective technique to prioritize test cases," *IEEE Trans. Softw. Eng.*, vol. 42, no. 10, pp. 918–940, Oct. 2016.

[46] Z. Li, Y. Bian, R. Zhao, and J. Cheng, "A fine-grained parallel multi-objective test case prioritization on GPU," in *Proc. Int. Symp. Search Based Softw. Eng.*, 2013, pp. 111–125.

[47] A. Arrieta, S. Wang, G. Sagardui, and L. Etxeberria, "Test case prioritization of configurable cyber-physical systems with weight-based search algorithms," in *Proc. Genetic Evol. Comput. Conf.*, 2016, pp. 1053–1060.

[48] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," in *Proc. Int. Conf. Softw. Eng.*, 2001, pp. 329–338.

[49] L. Inozemtseva and R. Holmes, "Coverage is not strongly correlated with test suite effectiveness," in *Proc. ACM 36th Int. Conf. Softw. Eng.*, 2014, pp. 435–445.

[50] T. T. Chekam, M. Papadakis, Y. Le Traon, and M. Harman, "An empirical study on mutation, statement and branch coverage fault revelation that avoids the unreliable clean program assumption," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng.*, May 2017, pp. 597–608.

[51] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, L. P. Cáceres, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Sep. 2016.

ANU BAJAJ received the B.Tech. and M.Tech. degrees in computer science and engineering from the Guru Jambheshwar University of Science and Technology (GJU S&T), Hisar, Haryana, India, in 2012 and 2014, respectively, where she is currently pursuing the Ph.D. degree in computer science and engineering. She was a Junior Research Fellow with GJU S&T, from 2016 to 2018, where she is a Senior Research Fellow. Her current research interests are within software engineering topics and include software testing, software maintenance and evolution, search-based software engineering, computational intelligence, and soft computing.

OM PRAKASH SANGWAN received the M.Tech. degree (Hons.) in computer science and engineering and the Ph.D. degree in computer science and engineering from the Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India, where he is currently a Professor with the Department of Computer Science and Engineering. He was the Deputy Director with the Amity Resource Centre for Information Technology and Head, Cisco Regional Networking Academy, Amity Institute of Information Technology, Amity University, Uttar Pradesh. He was an Assistant Professor with the Department of Computer Science and Engineering, School of Information and Communication Technology, Gautam Buddha University, Greater Noida, Uttar Pradesh. He is also a CISCO Certified Network Associate (CCNA) and a CISCO Certified Academic Instructor (CCAI). He has number of publications in international and national journals and conferences. His research interests include software engineering focusing on planning, designing, testing, metrics, and application of neural networks, and fuzzy logic and neuro-fuzzy. He is a member of Computer Science Teacher Association (CSTA), New York, USA, International Association of Engineer (IAENG), Hong Kong, and International Association of Computer Science and Information Technology (IACIST), USA, a professional member of Association of Computing Machinery, IEEE, and a Life Member of Computer Society of India, India.