EVALUATING THE IMPACT OF PRIORITIZING TEST CASES EXPLOITING GENETIC ALGORITHM

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF DEGREE
OF

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by:
MONIKA
(2K19/CSE/14)

Under the supervision of:

DR. SHAILENDER KUMAR
(ASSOCIATE PROFESSOR, CSE)
Department of CSE, DTU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering) Bawana
Road, Delhi – 110042

JULY 2021

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CANDIDATE'S DECLARATION

I, Monika, Roll No. 2K19/CSE/14 student of M.Tech. (Computer Science and Engineering), hereby declare that the Dissertation titled "Evaluating the impact of Prioritizing Test Cases exploiting Genetic Algorithm" which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: Delhi **MONIKA**

Date: 27th July, 2021 2K19/CSE/14

COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CERTIFICATE

I hereby certify that the Dissertation titled "Evaluating the impact of Prioritizing Test Cases exploiting Genetic Algorithm" which is submitted by [Monika], Roll No. 2K19/CSE/14 [Computer Science and Engineering], Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi **(DR. SHAILENDER KUMAR)**

Date: 27/07/2021 **SUPERVISOR**

ABSTRACT

It is essential to maintain the software on time by updating the code based on the changes taking place in the user requirements or technology. Test case prioritization is the regression testing technique that arranges the test cases during the software maintenance stage but it is not feasible to re-run every test case. Genetic algorithm, a search-based algorithm prioritize the test cases successfully and gives optimal outcomes. This paper gives a systematic literature review by prioritizing test cases using genetic algorithm by selecting the most relevant studies with the help of search strategy, quality assessment, data extraction and data synthesis process. We then explored the current research trend of studies with detailed analysis by covering every aspect of existing research. It is observed that the use of appropriate parameters, databases and tools can enhance the results. This review also yields limitations and suggestions for future researchers to understand and enhance the effectiveness of genetic algorithm by prioritizing test cases.

ACKNOWLEDGEMENT

I am very thankful to Dr. Shailender Kumar (Associate Professor, Department of Computer Science & Engineering) and all the faculty members of the Department of Computer Science at DTU. They all provided me with immense support and guidance for the project.

I would also like to thank the University for providing the laboratory, infrastructure, test facilities and environment so that I can work without obstacles.

I would also like to thank our lab assistants, seniors, and peer groups for providing me with all the knowledge on various topics.

MONIKA

(2K19/CSE/14)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SDLC	**S**oftware **D**evelopment **L**ife **C**ycle
TCP	**T**est **C**ase **P**rioritization
GA	**G**enetic **A**lgorithm
SLR	**S**ystematic **L**iterature **R**eview
SUT	**S**oftware **U**nder **T**est
LOC	**L**ines of **C**ode
APFD	**A**verage **P**ercentage of **F**ault **D**etection
APSC	**A**verage **P**ercentage of **S**tatement **C**overage
APBC	**A**verage **P**ercentage **B**lock **C**overage
APDC	**A**verage **P**ercentage **D**ecision **C**overage
FB	**F**ault-based
CB	**C**overage-based
RQ	**R**equirement-based
HB	**H**istory-based
MF	**M**odification-based
GB	**G**enetic-based
CP	**Composite**
NSGA-II	**N**on-dominated **S**orting **G**enetic **A**lgorithm II
MOEAs	**M**ulti-**O**bjective **E**volutionary **A**lgorithm

HCSs	**H**ighly-**C**onfigurable **S**ystems
CPSs	**C**yber-**P**hysical **S**ystems
HV	**H**ypervolume
STIPI	**S**earch-based **T**est **C**ase **P**rioritization
BRKGA	**B**iased **R**andom-Key **G**enetic **A**lgorithm
ASAGA	**A**daptive **S**imulated **A**nnealing **G**enetic **A**lgorithm
ETS	**E**pistatic **T**est **C**ase **S**egment
HGA	**H**ypervolume-based **G**enetic **A**lgorithm
CBGA-ES	**C**luster-based **G**enetic **A**lgorithm with **E**litist **S**election
PMX	**P**artially **M**apped **C**rossover
PUX	**P**arameterized **U**niform **C**rossover
SPLOT	**S**oftware **P**roduct **L**ines **O**nline **T**ools
OS	**O**perating **System**
RAM	**R**andom **A**ccess **M**emory
FDC	**F**ault **D**etection **C**apability
APEC	**A**verage **P**ercentage of **E**lement **C**overage
PCA	**P**rincipal **C**omponents **A**nalysis

CHAPTER-1

INTRODUCTION

As the software industry is proliferating, reliability and customer satisfaction are the primary goals for software developers [1]. Therefore, software maintenance is necessary in the software development life cycle (SDLC) that updates the software according to a change in user requirements, diverse technology, recurrent updates, and browser compatibility to be used in the long term. The maintenance cost of the software can account huge portion of the overall software cost [2].

Regression testing helps to validate the updated software [3]. Software developers often preserve the recently generated test suites and reuse those test suites for subsequent regression testing [4]. Such testing is used after modifying the software to guarantee that the earlier software parts are working fine [5] and the extended source code works as specified with no new bugs or defects introduced [6]. It is retesting of a system where both existing and newly developed test cases are executed. Regression testing may take much time while performing all the test cases [7]. One example of the literature shows that regression test suites take approximately thousand hours to complete thirty thousand test cases [8].

Yoo and Harman [9] provides the optimization techniques and conducted a survey on regression testing approaches i.e., test case minimization, selection and prioritization. Test case minimization technique identifies the monotonous experiments and permanently eliminate unnecessary test cases to decrease the test suite size [10]. However, the removal of test cases can decrease the potential of detecting the faults of the test suite. Test case selection lessen the test cases by choosing some test cases based on some measures to be executed without deleting the test cases. It recognize those test cases essential to the latest modification of the source code [11]. Test case prioritization (TCP) arranges the test cases dependent on favored parameters that provide early optimization [12]. TCP focuses on arranging the test cases with some criteria that may identify faults at an early stage or increase fault detection rate, finding the high-risk faults earlier, improving the probability of uncovering the mistakes identified with explicit code changes prior, or to gain the faith in the unwavering quality of the framework [13]. TCP determine the final arrangement of a series of test cases and is performed suitably [9]. Test prioritization helps to decrease time and amount to sustain service-oriented applications. The software TCP improves the viability of the test in software testing. It is observed that test case minimization and selection may skip essential test cases causing a high chance of unknown errors in the software, whereas, TCP covers every test case by picking the best test case execution sequence, increasing the fault detection rate [14]. It also provides reliable and cost-efficient outcomes [8]. TCP is helpful because tests accumulate over various corrections and system versions [15]. Researchers have presented numerous methodologies to ameliorate the performance of regression testing to make it cost-efficient. Some techniques have been examined and evaluated by many researchers [7, 11, 16, 17, 18, 19, 20, 21, 22] but nobody has provided the detailed conclusion. Therefore, we have selected TCP over other regression testing techniques.

Various researchers have applied several different strategies to serve the regression testing process with linear programming, greedy algorithms etc. Due to the speculative behaviour of regression testing, traditional methods are incapable to optimize the solution. These constraints imply the use of nature-inspired techniques having the potential to achieve solution using limited resources. Such algorithms are becoming very famous because of clarity, flexibility, derivation-free implementation and abilities to eliminate the local-optima than other optimization procedures [23]. Due to the above reason, the researchers started working towards nature-inspired algorithms to select the optimal solution for a current problem. There are many nature-inspired algorithms used in TCP [24]. Genetic algorithm (GA) is one of the nature-inspired algorithm that describes the "Survival of the fittest" theory [14], introduced in 1975 by John Holland [25] with the book name "Adaptation in Natural and Artificial Systems". Further GA was studied by many authors like Goldberg [26] and De Jong [27]. It is the most famous evolutionary algorithm [28]; due to expansion in prominence, GA is used in software development by the software industry [29, 30, 31]. Though it is very effective, but sometimes it gives local optimal solutions during complex problems [32].

To provide a broad aspect, a systematic literature review (SLR), also known as secondary study is conducted to collect and compare these techniques on various parameters. SLR is a meticulous investigation of research evidence [33]. It helps to provide a detailed summary of current proofs using research questions. This paper gives a SLR to review all the existing TCP techniques using GA present to date. It has covered various parameters like: TCP techniques, types of GA, operators used in GA, tools, running environment, programming language, comparisons, goals, limitations and suggestions in TCP using GA to help the new researchers understand this domain. We have also provided detailed knowledge on regression testing, TCP, GA and TCP using GA.

1.1 BACKGROUND AND MOTIVATION

Regression Testing is used during the maintenance stage when Software under test (SUT) is improved and the software industry frequently uses it to validate the modified software version. It guarantees that the product still performs well after being modified. There can be many reasons for software being updated: adding new features or functionality in the software, change in user requirements, adaptation to a new environment, modification in code due to requirements and correction in the software system. Regression testing gives faith to the software developer that the modified software will run fine without having harmful effects on the remaining parts of the code [34, 35]. This testing is costly as it ensures the best quality software during the maintenance stage [34, 36, 37] and execution of every test case during regression testing is time-consuming [38]. Elbaum et al. [11] performed regression testing to validate the software modifications and found it expensive. They also observed that running software took seven weeks, having only twenty thousand lines of code (LOC) and an extensive software system took thirty days after running tests regularly. TCP helps to provide effectiveness in terms of both cost and time used in regression testing.

Test Case Prioritization is the positioning of the test cases where the test cases with high priority executes first [39]. It checks for all the possible permutations of the test suite; therefore, it is an NP-Hard problem [40]. Once the software is deployed in the market, it undergoes into maintenance stage for ten to fifteen years. The prioritization is done because the essential test cases will get the chance to execute first. It also removes the possibility of duplicate test cases, hence reducing the testing time complexity [41]. The goal is to run the test cases based on detecting faults earlier [42]. TCP can be distinguished as a single and a multi-objective optimization problem [6]. The cost of retesting software at regular intervals can be high; therefore, trivial and anomalous test cases are removed to reduce the test suite with economical amount [43]. Testers categorize between high and low priority test cases to perform reliable and efficient execution during TCP. Thus, gives good quality software to the customers and keeps the faith in the company as well. Hence, TCP is more effective than other regression techniques known as test case minimization and selection and TCP is becoming more prevalent [9]. TCP is further classified into various techniques like coverage, modification, fault, requirement, history, search and risk-based technique [44, 45]. The role of these techniques as well as limitations were discussed by Khatibsyarbini et al. [46].

The coverage-based technique increases the fault detection rate if the test suite is achieving more coverage; therefore, the goal is to increase coverage of various program modules such as statement, branch and methods by the test case. Rothermel showed that high coverage is proportional to a high fault detection rate [7, 47]. Rothermel also proposed new coverage-based techniques contingent on total branch, total statement, additional branch and additional statement coverage [12]. The history-based technique allocates priority to the test cases according to some historical records obtained from execution of the test case and it was also explored by [4, 48, 49]. Kim and Porter [50] first proposed the concept of a history-based approach in TCP. The test cases that cover more program modules like program statements, functions based on the previous testing will be assigned more priority [4]. The requirement-based technique is according to the user's demand to prioritize the test cases. Srikant [51, 52] proposed the system level approach known as Prioritization of Requirements for Testing: PORT V1.0 in TCP. The modification-based technique is prioritizing the test cases based on the modifications done in the program. Fault-based techniques prioritize the test cases dependent on the quantity of faults covered. The search-based technique is searching over the optimal or close to an optimal solution in TCP [40]. The risk-based technique is the probability of software at risk. There are many more techniques in TCP, but it was noticed that most researchers consider fault or code coverage and execution cost in the genetic algorithm [24].

Researchers use various metrics to check the performance of the system. The Average Percentage of Fault Detection (APFD) was suggested by Elbaum et al. [13] which identifies the fault detection rate of a test suite based on the technique used for prioritization. The Average Percentage of Statement Coverage (APSC) [14, 53, 54] counts the maximum number of modified statements. Average Percentage Block Coverage (APBC) and Average Percentage Decision Coverage (APDC) were also used by Li et al. [14]. It was observed that APFD, total coverage, implementation cost and low percentage of test suite size are very popular metrics used to evaluate the performance [24].

Genetic algorithm is an evolutionary search-based heuristic methodology used for optimization, also known as a population-based technique. It includes the natural selection of fittest individuals to make offspring of the upcoming generation. Selection, crossover and mutation are the GA operators that helps to produce feasible solutions [38]. GA is used in many areas because it is efficient [55]. It is known for solving complex problems with high time complexity, NP challenging problems like 0/1 knapsack and travelling salesman problem to provide an optimal solution. GA starts with the selection process of individuals followed by reproduction phase where new offspring is generated and parents for the next generation are chosen. New chromosomes are generated using crossover and mutation operators by switching the genes of the selected parents. The steps are repeated until they reach the stopping condition [56]. GA is more popular than other nature-inspired algorithms and is primarily used in regression testing [24].

TCP is the reordering of the test cases and is efficacious when it should have some termination condition to not go into an infinite loop. The termination condition depends on population trends [55]. Hence, GA provides the time constraint and optimized solution to a given problem to find maximum faults. GA in TCP provides the optimized results than the other algorithms [57].

The process starts with the creation of the initial population of random chromosomes/individuals. The population experience the fitness test according to the coverage criteria for prioritizing the test cases. The individuals should contain the information about solutions, therefore, the chromosomes must be encoded. There are various encoding schemes like permutation, binary and tree encoding. Depending on a fitness function, the selector operator selects the best test suites with a high fitness value. There are various selection mechanisms like rank-based, roulette wheel, tournament and truncation selection. With the help of the selection mechanism, the next generation or new offspring is generated and the fitness is evaluated for the newly generated chromosomes. To provide diversity to the newly created offspring, crossover and mutation operators are used. The process continues till the fittest offspring are obtained and met the termination condition. Fig. 1.1 shows the flow of GA and a variety of encoding schemes, selection, crossover and mutation operators.

The crossover operator helps in generating one or more than one offspring from the parents, generally two parents. In a single-point crossover, only one point is chosen randomly as a crossover point and all the data is swapped among the two parents beyond the crossover point chosen. Two-point crossover is the technique where two crossover points are considered and beyond the crossover point, the data is exchanged between the parents. In a partially mapped crossover, the genes are exchanged between the two points that are selected from the two parents, while the remaining genes are filled by partial mapping. In an ordered crossover, two points are selected where the segment value is copied to the offsprings as it is and after the second cut point, the values of one parent are replicated to another parent in the exact order neglecting the duplicate values. Cycle crossover detects the cycle between the two parents [28]. Tree crossover uses two points for two parents respectively to separate them and new offspring is generated after exchanging the below crossover point [56]. Average crossover generates one offspring by an average of two parents [38].

To achieve a global optimum solution, a mutation operator helps to provide genetic diversity in the newly created offspring at a higher level and helps remove duplicate test cases. Various mutation operators are swap mutation [28, 58], insertion mutation [38], addition/removal operator, substitution operator etc. In swap mutation, two points are selected from the chromosome at random and interchanged while keeping other positions unchanged. Inversion mutation is based on gene position by selecting one gene randomly from the chromosome and then rearranging the positions of the other gene by re-inserting the selected gene at the new random position with minimum mutation probability. An addition/removal operator is used for adding or removing a test case randomly at a random index position. The substitute operator substitutes the test case randomly with some other valid test case.

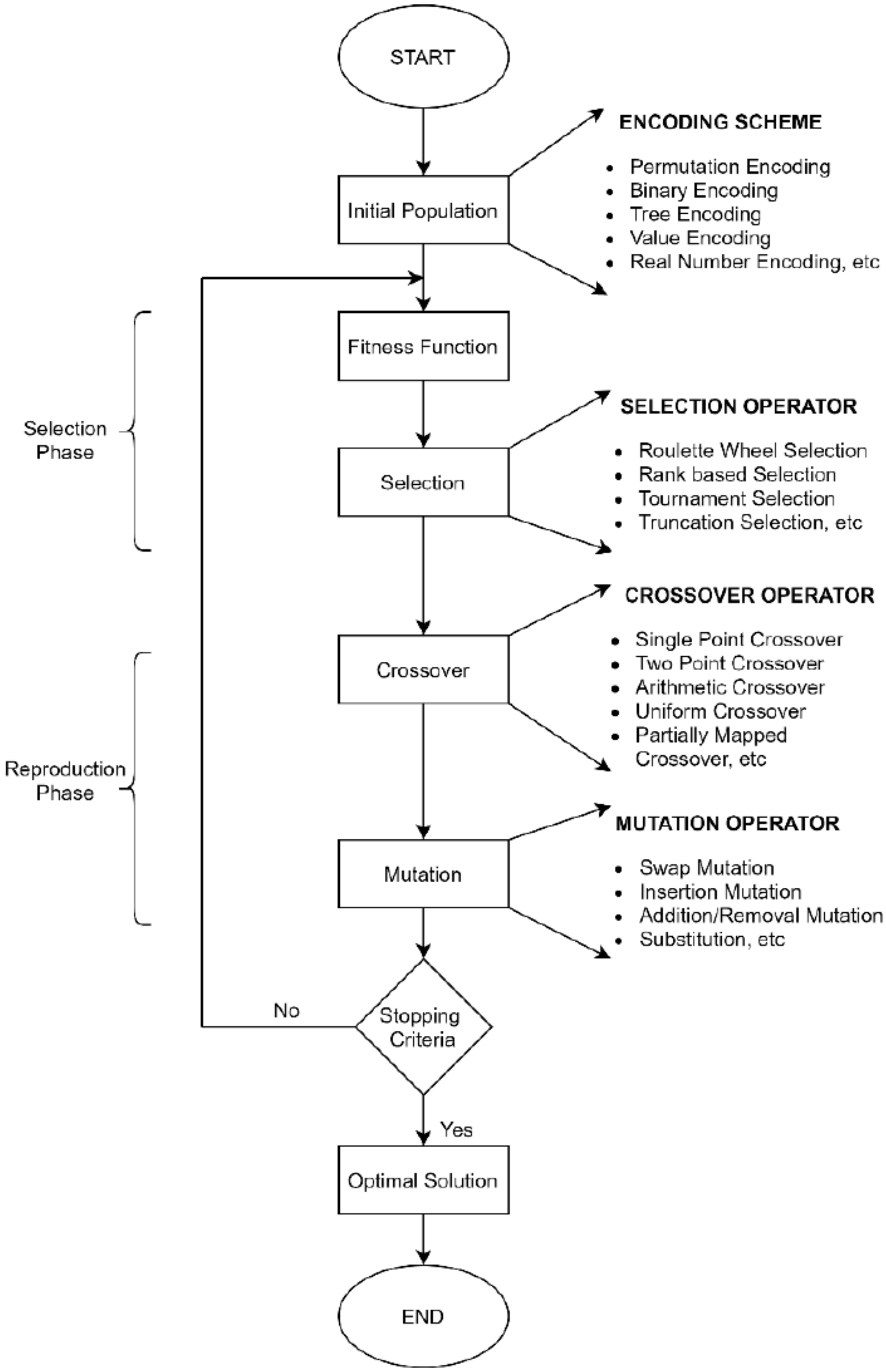

Fig. 1.1 Process of genetic algorithm with various encoding schemes and operators

The motivation of this research work arrives from the need of a systematic literature review that can help the software practitioners to prioritize the test cases with the help of genetic algorithm in an efficient manner. This will also guide the researchers in determining appropriate tools, techniques, methodologies commonly adopted by researchers, limitations and suggestions of present literature that help in achieving better software quality within budget in timeliness manner.

1.2 RESEARCH OBJECTIVE

The goal of this research work is to provide a systematic literature review to answer questions related to effect of genetic algorithm in prioritizing the test cases (complete process and research questions addressed are discussed in detail in Chapter 3 and 4 respectively) and then provide suggestions to prioritize the test cases efficiently using genetic algorithm.

1.3 PROPOSED WORK

To conduct the systematic literature review, 35 primary studies from January 1999 to March 2021 were selected from premier journals and reputed conferences that apply GA on prioritizing test cases to evaluate their effects on software quality attributes. We have covered overall 21 questions to address the trend of GA in TCP. Papers are selected from electronics databases i.e. IEEE Xplore, SpringerLink, Science Direct (Elsevier), ACM and Wiley with the use of sophisticated search string using boolean ANDs and ORs followed by inclusion-exclusion criteria. Quality questionnaire is formed with 12 quality questions to choose the most relevant papers. Data extraction helped to extract the most appropriate data using data collection form. Finally, data synthesis process helped to summarize the results using various visualization methods like bubble charts, bar charts, line graphs and tables evaluation.

1.4 ORGANISATION OF THESIS

The rest of the thesis is divided into following Chapters.

Chapter 2 discusses the related work in the field applying genetic algorithm in prioritizing the test cases to improve software maintenance phase. **Chapter 3** explains the research methodology adopted in the current research work. **Chapter 4** answers the research questions imposed at the start of the research work and discusses the results in detail. **Chapter 5** discusses threats to validity in the end; **Chapter 6** concludes the research work with inferences drawn from analysis and gives directions for future work.

CHAPTER-2

RELATED WORK

TCP is a prevalent and primarily used approach in regression testing. GA provides the optimal results and hence researchers started using GA in TCP. Yoo and Harman [9] published a survey about techniques of regression testing known as test case selection, minimization, prioritization and discussed it's applications, usage followed by the challenges. They observed that the regression testing in a controlled manner would prioritize the test cases depending on the rate of fault detection in TCP. APFD metric was used to identify the fault detection rate which means that the high value of APFD will provide a fast fault detection rate. They provide a detailed survey in TCP from the year 1999 to 2009 and observed the increasing trend of TCP. They explored only two studies of TCP using GA, first by Li et al. [14] and second by Walcott et al. [59].

Singh et al. [60] presented a systematic review for TCP. They presented the TCP techniques as fault-based (FB), coverage-based (CB), requirement-based (RQ), history-based (HB), modification-based (MF), genetic-based (GB), composite (CP) approaches and many other techniques. They covered the year of publications from 1997 to 2011. They compared 65 research papers, out of which 106 new prioritization techniques were found in 49 papers and 16 were comparison based. The highest number of publications was found in the year 2007 and 2008, covering 11 techniques each. The maximum number of techniques discussed was 21 in the year 2009. The coverage-based technique was the first technique to be used in 1997 and it was also noted that 44% of the coverage-based techniques were more frequently used. Other TCP techniques like fault-based, history-based, requirement-based were introduced during the year 1999, 2002, 2005, respectively. TCP using GA was documented in the year 2006. Mostly referred granularity utilized by 106 techniques were system level (20%), web services (17%), statement level (14%) and function level (13%) granularity. 70% of the input method was based on source code in TCP techniques. It was noticed that 48% of L. Industrial Language (LInd) was the commonly used language by the researchers, various tools were also studied like Aristotle, Sofya, Emma and Unix based tools. Also, the size of the artifacts was majorly in KLOC's. They also discussed various metrics such as APFD, APFDc, APFD alike and

Bonferroni. It was observed that no such technique was studiedthat can expand the fault detection rate in TCP. Only Rothermel et al. [12], Elbaum et al. [16] and Rothermel et al. [19] were better to "additional-fault exposing potential" in comparison with others. Only two papers were covered in their study of TCP using GA as [59, 61] and concluded that GA is a new upcoming area for TCP.

Catal and Mishra [62] presented a systematic mapping by covering 120 papers, highlighting the techniques used in TCP, improvement to those techniques and the current trends in TCP. Only 16 papers were used to compare various prioritization techniques. They covered the period from 2001 to 2011. They showed that after the year 2006, the researchers' interest has increased in TCP and also discussed TCP techniques suggested by Yoo and Harman [9]. In their study, coverage-based technique (40%) was majorly used by the studies. 18 new research topics were identified, where 59% is the generation of prioritizing methods, 12% is for comparison, 11% for performance measures and 18% for others. The classification of papers was based on various parameters like the development of the prioritization method (62%), experiment (27%), review (3%), theory (5%), own experience (2%) and survey (1%). 64% out of the total percentage were based on industrial projects. They also discussed the evaluation metrics, APFD (34%) was commonly used metric and others were TPFD, ASFD, NAPFD etc. They classified the datatype set as public, private, partial and none. The genetic algorithm in TCP was not covered in this research paper.

Khatibsyarbini et al. [46] gave a SLR on various techniques used in TCP. They evaluated 80 papers as a preliminary study. They reviewed from 1999 to 2016 and showed that TCP is very popular in regression testing. They also discussed various techniques of TCP as fault, coverage, requirement, risk, search, history-based and others, including cost-aware, Bayesian-Network-based techniques etc., where the search-based approach was found to be the most appropriate method among the other studies. They also discussed the evaluation metric, APFD, APFDc, CE and others, where APFD (51%) was found as most commonly used metric. Various artifacts were also studied like real case studies, siemens suite, space programs, TSL and ATM. Few papers were also covered regarding GA in TCP techniques. It was concluded that many TCP techniques are already available that can be improved to give importance to early prioritization.

Anu Bajaj and Om Prakash [63] presented the secondary study on TCP using GA and reviewed 20 papers from 1999 to 2018. Their study shows the positive growth in

TCP from the year 2014 onwards. They classified the research methodology as an experiment-based, case study, development and empirical. Various TCP coverage based techniques like fault, code, requirement, configuration and others were also covered. The distribution of studies were based on open-source programs and industrial programs. Program size was classified as small, medium, large and extensive programs. They also described the tools and various types GA used in TCP along with other parameters. They discussed five metrics, APFD, FDC, APEC, CE and others. It was shown that GA is capable of solving TCP problems.

Only a few papers are based on a secondary study that discussed the role of GA in TCP with limited parameters. Hence, many more uncovered parameters can be added to the secondary study to make it more useful for future researchers. Also, there is only one recent study that has given a SLR and discusses the trend of GA in TCP by covering only 20 studies until August 2018. Our work is unique because we have included more papers with recent studies and have covered more parameters in a broader sense to provide detailed knowledge towards GA in TCP.

CHAPTER-3

RESEARCH METHODOLOGY

SLR aims to gather and study the existing research for a particular research topic with various distinct activities. It provides honest, precise and auditable information that helps in identifying the research trend. We followed the guidelines given by Kitchenham [64] that describe three stages of SLR: Planning, Conducting and Reporting as given in Fig. 3.1. Thus, a SLR provides detailed information that helps in understanding the latest trend in that area.

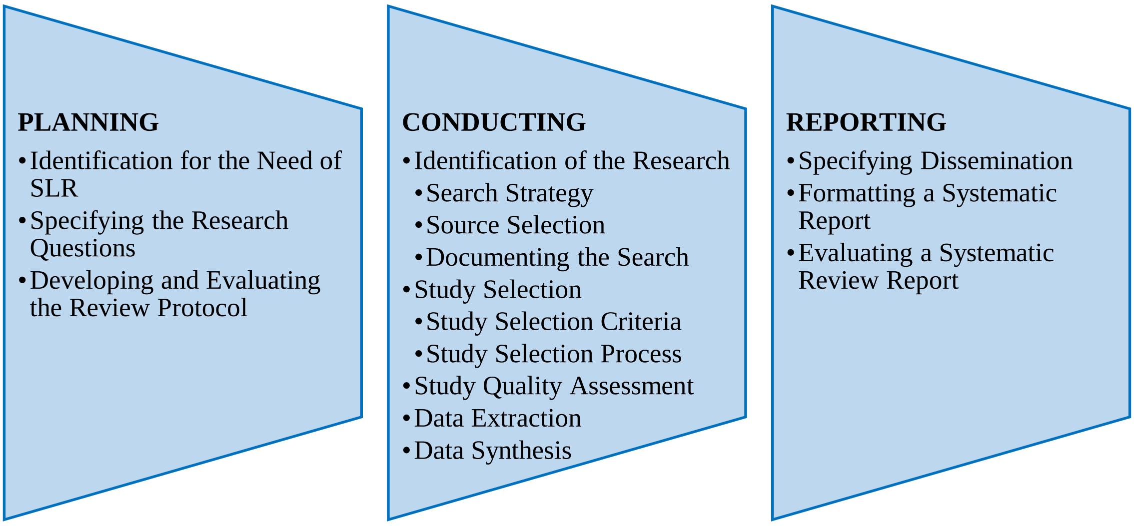

Fig. 3.1 Phases in SLR

3.1 Planning a SLR

It is requisite to understand the need for review in a particular area before conducting a SLR. The stages in the planning of SLR include research questions that SLR addresses, the plan for review procedures and the evaluation process subject to independent markings.

3.1.1 Need for Systematic Review

A SLR provides in-depth information by combining all the existing information of specific research. With the advancement in technology, it was noted that the trend of GA in TCP is yet uncovered. GA in TCP is becoming popular as it provides promising results, therefore, we have collected and synthesized research evidence on various topics from relevant studies.

3.1.2 Specifying the Research Questions

We have developed 6 questions and further classified them into 21 sub-questions to explore most of the parameters in GA using TCP. Table 3.1 shows the research questions with their respective motivations. All the research questions are inter-connected with each other and are helping to answer the extra findings in a SLR. We have used Petticrew and Roberts guidelines to set the research questions [65]. The guidelines are based on PICOC standard i.e., Population, Intervention, Comparison, Outcomes and Context. The criteria and scope of research questions is shown in Table 3.2.

TABLE 3.1 Research questions with their respective motivations

RQ#	RESEARCH QUESTIONS	MOTIVATION
1	What is the research trend of GA in TCP?	
1.1	What is the distribution of studies in terms of publication year?	To characterizing the latest domain of TCP using GA. It includes year of publication of papers, distribution of electronic database along with type of paper used and TCP methods used by the researchers.
1.2	What is the publication trend?	
1.3	Which type of TCP technique is used?	
2	How GA is used in TCP for ordering of the test cases?	
2.1	Which type of GA is used?	To investigate various attributes of GA in terms of population size, generation size and various operators of GA required for research.
2.2	How population is represented in terms of encoding scheme?	
2.3	What is the population size?	
2.4	What is the generation size?	
2.5	Which selection operator is used?	
2.6	What type of crossover operator is used?	
2.7	What is the crossover probability/rate?	
2.8	What type of mutation operator is used?	
2.9	What is the mutation probability/rate?	
3	What is the testing environment of GA in TCP?	
3.1	Which tools are used to perform the experiment?	To investigate the tools required, running environment i.e., OS, RAM and Processor used and the type of programming language used for implementation.
3.2	What is the running environment?	
3.3	Which programming language was used for implementation?	
4	How were these approaches validated?	

TABLE 3.1 Research questions with their respective motivations (continued)

RQ#	RESEARCH QUESTIONS	MOTIVATION
4.1	What are the evaluation metrics used for evaluation?	To find out whether the outcomes are analysed properly and reliable enough as compared to other techniques.
4.2	Whether the technique has been compared with other techniques?	
4.3	What statistical techniques are adopted by the researchers?	
5	Whether the current technique is effective with regard to execution time and cost?	Find if current study is effective regarding time and cost.
6	What is the significance of GA in TCP?	
6.1	What are the goal and limitation of the current research?	To find out the reason for applying GA and how it is being applied in TCP. Also, to find out the shortcomings of using GA in TCP and provide recommendations (if any).
6.2	Is there any key issues and gaps in existing research that can be addressed in future?	

TABLE 3.2 Criteria and scope of research questions based on PICOC guidelines

CRITERIA	SCOPE
Population	The writing on the utilization of GA in TCP
Intervention	TCP technique using GA
Comparison	A Holistic comparison to study the trend of current research
Outcomes	Various parameters in TCP using GA with evidences to help future researchers
Context	Review(s) of studies in TCP using GA

3.1.3 Develop and Evaluate Review Protocol

The review protocol provides the techniques to manage a SLR on a particular topic. A pre-established protocol helps to minimize the probability of the researcher being biased. The protocol components include reviewing elements with some additional information like background, selecting research questions, strategy to search primary studies, study selection criteria, study selection procedures, quality assessment, data extraction technique and data synthesis of the extracted data. Fig. 3.2 shows the stages in the review protocol.

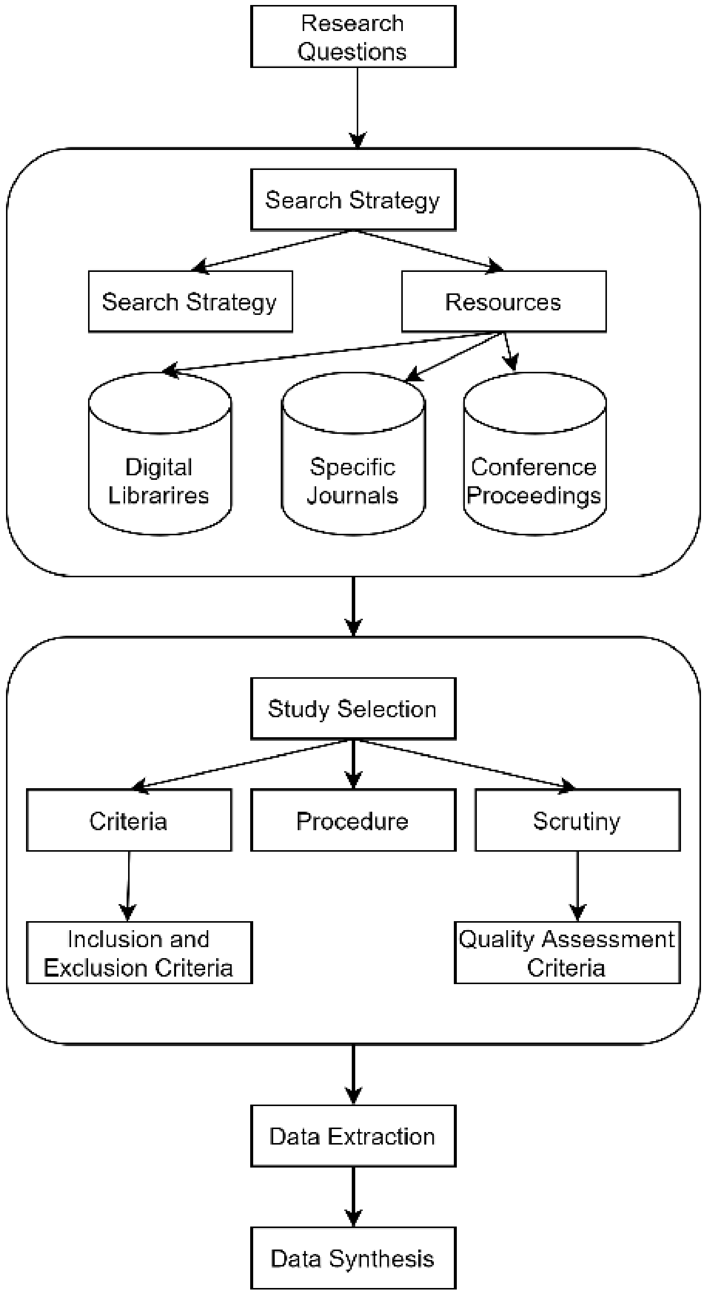

Fig. 3.2 Developing a review protocol

3.2 Conducting a Systematic Literature Review

Once the protocol is set, the later stage is the review paper. This phase is further divided into five stages: identification of research, study selection, study quality assessment, data extraction and data synthesis. All the stages are discussed below.

3.2.1 Identification of Research

The goal is to detect substantial primary studies pertinent to the research questions. Search strategy and source selection are two fundamental approaches used to select the primary studies. It is required to find and follow the search strategy. After specifying the research questions, a search strategy identifies search terms and key sources to be explored from where primary studies are captured. The search technique gives the pertinent studies to answer the research questions. Electronic databases are used for searching and extraction of studies. For this SLR, Table 3.3 shows the search strategy that are merged using Boolean connectors ORs and ANDs as suggested by Buckley et al. [66].

TABLE 3.3 Primary search strategy

Derive search keywords	Based on research questions and PICOC criteria
Determine synonyms, abbreviations and alternate spelling	For example, test and testing; test case and test suite; prioritization and optimization; genetic algorithm and search based heuristic technique and population based technique; test case prioritization and regression test prioritization.
Usage of sophisticated search strings	It includes Boolean connectors i.e., ORs and ANDs. 1) Boolean OR is used to incorporate alternative spelling and synonyms. 2) Boolean AND is used for linking major search keywords.

We have used the online repositories for literature source selection i.e., IEEE Xplore, SpringerLink, Science Direct (Elsevier), ACM and Wiley using the search string. Table 3.4 shows the list of selected electronic databases to access relevant primary studies.

TABLE 3.4 List of electronic databases explored

S. No.	E-Resource	Link to Access
1	IEEE Xplore	ieeexplore.ieee.org/
2	SpringerLink	http:///www.springer.com/in
3	Elsevier ScienceDirect	www.sciencedirect.com
4	ACM Digital Library	www.acm.org/
5	Wiley Online Library	https://onlinelibrary.wiley.com/

SLR must be clear and reproducible without any ambiguity. The information provided in the review and search used for source selection should be transparent to future researchers. The search was done using search string on articles from 1999 to March 2021. Initially, the search was based on titles, abstracts, keywords and bibliographic information. Studies that are found useful in the reference section were also selected. We have created the search string as, (((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"))).

3.2.2 Study Selection

Once the studies are collected, the study selection stage helps to pick specific primary studies that can cover all the parameters based on the research questions. To reduce bias-likelihood, a selection criterion is established at the time of research protocol definition. All the primary studies were collected in full text. Later it went to inclusion and exclusion criteria represented by Fig. 3.3. Testing and verification for accuracy were achieved following the guidelines suggested by Brereton et al. [67] for inclusion and exclusion criteria. The quality of chosen papers was also identified to capture their relevance regarding research questions.

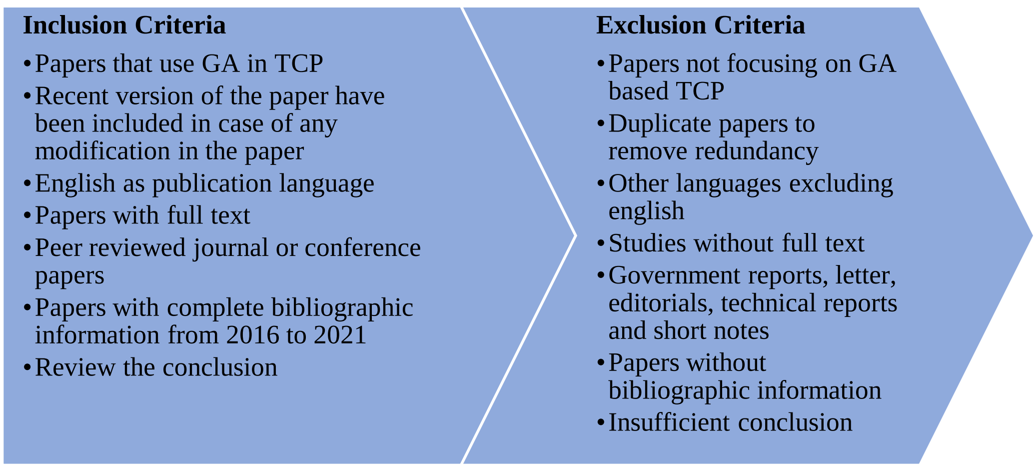

Fig. 3.3 Inclusion and exclusion criteria

The selection method is a multistage process as discussed below.

Stage 1: Selection of studies in a liberal manner considering titles and abstracts. Initially, we found 522 studies by applying the search string strategy. We found 259 studies from IEEE, 118 studies from Springer, 9 studies from Elsevier, 106 studies from ACM and 30 studies from Wiley. Later on, screening of studies was done based on full text read to select only relevant studies.

Stage 2: It includes examination of the references, citations and the conclusion of selected studies.

Stage 3: We found 35 primary studies for a SLR based on detailed quality criteria. Fig. 3.4 describes the search and selection process for selecting the primary studies.

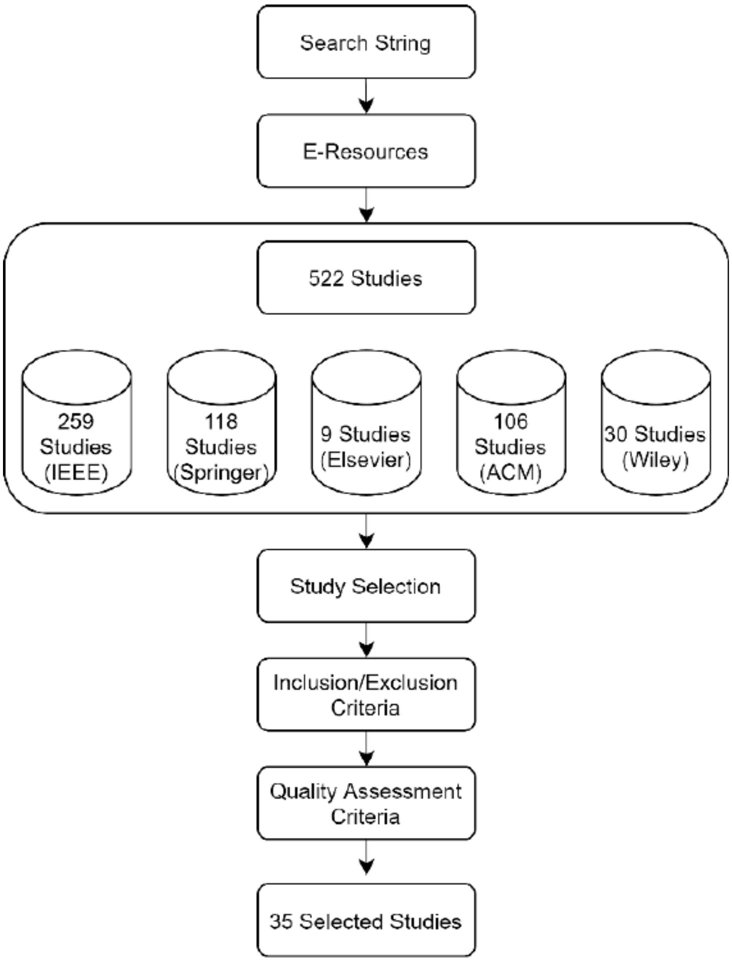

Fig. 3.4 Search and selection process

3.2.3 Study Quality Assessment

The quality questionnaire is formed after inclusion-exclusion criteria as it helps in selecting the most appropriate papers. It includes 12 quality questions to assess their significance and provides weights to the studies to the research questions addressed in SLR. 3 parameters were used i.e., 0 for NO, 0.5 for Partial and 1 for YES. We consider a threshold score of 7.5 to ensure the capability of addressing each research question. Subsequently, many research papers were excluded whose quality assessment was less than 7.5. Finally, 35 papers were selected for evaluation of primary studies and we have allotted paper ID to the selected primary studies for better understanding. Table 3.5 describes the quality assessment questions and table 3.6 represents the quality assessment score of primary studies with their respective paper ID and paper reference. For each primary study, the rank for each question is summed to calculate the final score.

TABLE 3.5 Questions of quality assessment

QA#	Quality Assurance Questions
1	Is the goal of the primary study stated clearly?
2	Is GA in TCP defined appropriately?
3	Is the proposed technique clearly described?
4	Is the process of data collection carried out properly?
5	Is the shared data sufficient to support results and conclusions?
6	Are the results validated properly?
7	Is the derived technique compared with other techniques?
8	Are the data sets described properly?
9	Are the statistical methods expressed?
10	Are the statistical methods justified?
11	Do the results support the conclusion?
12	Are validity threats discussed?

TABLE 3.6 Quality assessment score

Paper ID	Paper Reference	Quality Assessment Score
P1	Huang at el. [4]	11
P2	Khanna at el. [6]	12
P3	Li at el. [14]	11.5
P4	Carballo at el. [28]	11
P5	Wang at el. [29]	8.5
P6	Wang at el. [31]	11.5

Paper ID	Paper Reference	Quality Assessment Score
P19	Ahmed at el. [69]	9
P20	Li at el. [70]	9.5
P21	Ray and Mohapatra [71]	8
P22	Marchetto at el. [72]	10.5
P23	Bian at el. [73]	10
P24	Epitropakid at el. [74]	11.5

TABLE 3.6 Quality assessment score (continued)

Paper ID	Paper Reference	Quality Assessment Score	Paper ID	Paper Reference	Quality Assessment Score
P7	Yan at el. [32]	10	**P25**	Di Nucci at el. [75]	12
P8	Mishra at el. [38]	8	**P26**	Arrieta at el. [76]	10.5
P9	Yuan at el. [40]	9	**P27**	Parejo at el. [77]	12
P10	Azam at el. [41]	7.5	**P28**	Pradhan at el. [78]	11.5
P11	Yadav and Dutta [53]	8.5	**P29**	Arrieta at el. [79]	11.5
P12	Mishra at el. [54]	8.5	**P30**	Kanugo at el. [80]	10
P13	Jun at el. [55]	8	**P31**	Kumar and Sujata [81]	8
P14	Yadav and Dutta [56]	10	**P32**	Padmnav at el. [82]	9
P15	Mukherjee and Patnaik [58]	12	**P33**	Mekuria Habtemariam and Kumar Mohapatra at el. [83]	11
P16	Walcott at el. [59]	12	**P34**	Khanna at el. [84]	11.5
P17	Conrad at el. [61]	11	**P35**	Pradhan at el. [85]	11
P18	Masri and El-Ghali [68]	8.5			

3.2.4 Data Extraction

The data extraction process was executed by following steps:

Step 1: Once the quality assessment is complete, data extraction forms are prepared to extract relevant data based on various parameters. One author analyzed 35 primary studies to perform data extraction after quality assessment stage.

Step 2: The form of data extraction was filled for each of the primary studies. Another author, a prominent educator in Computer Science and Engineering, validated this data by investigating the primary studies on selected parameters. It helped in collecting all the pertinent details to answer the research questions imposed by a SLR.

Step 3: In case of any disagreement found, a meeting was conducted to validate the data using a separate form to mark and update the errors. The final data is saved in an excel spreadsheet to use this data as input for data synthesis. Table 3.7 shows the data collection form.

TABLE 3.7 Data collection form

Category	**Sub-Category**	**Additional Notes**
Bibliographic Information	a) Unique Identification Number b) Publication Title c) Publication Year d) Author e) Publisher f) Publisher Type g) Citation Count h) Reference Count i) Email	
Study Focus	a) Domain Topic b) Problems c) Scope d) Motivation e) Objectives	
Research Methodology	a) Case Study b) Survey c) Experiment d) Observation e) Questionnaire	
Test Case Priorotization Techniques	a) Coverage Based Technique b) History Based Technique c) Requirement Based Technique d) Modification Based Technique e) Fault Based Technique etc	
Genetic Algorithm	a) Type of GA b) Flow c) Population Size d) Type of Encoding Scheme e) Fitness Function f) Selection Mechanism g) Crossover Operator h) Crossover Probability i) Mutation Operator j) Mutation Probability k) Any other parameter	
Evaluation	a) Evaluation Metrics b) Validation Criteria	
Results	a) Observation b) Limitations c) Future Work	

3.2.5 Data Synthesis

Data synthesis summarizes the facts and Fig. s from the extracted data to acknowledge the research questions. The extraction form is synthesized in a very consistent way with the research questions in SLR. It is a meta-analysis procedure to classify the results by different researchers to reach a certain conclusion that answers the research questions. Different visualization mechanisms like bubble charts, bar charts, line graphs and tables are used for the evaluation.

3.3 Reporting the Review

The third phase includes meaningfully and analytically reporting the findings from a SLR along with limitations and future directions as discussed in Section 4.

CHAPTER-4

FINDINGS AND DISCUSSIONS

The final stage of a systematic review necessitates observing and expressing the outcomes of the research questions. The conclusions captured from the primary studies will be discussed and presented meaningfully. It can help the researchers to understand the latest trend in existing research.

4.1 Distribution of studies regarding publication year (RQ 1.1)

This section discusses the classification of selected studies with regard to year of publication from 2016 to 2020 as represented in Fig. 4.1. The use of GA in TCP was founded in 2006 when Walcott et al. [59] used GA in regression TCP technique to rearrange the test suites with time limitations. Li et al. [14] in 2007 compared GA with different algorithms like greedy, additional greedy, 2-optimal and hill-climbing algorithm and showed that GA helps in the ordering of TCP to add the next order while considering the entire orderings. As an addition to Li et al. [14], Conrad et al. [61] used a variation of mutation, crossover and selection operators and observed promising results. Henceforth, the researchers started working with TCP using GA. Ahmed et al. [69] designed an approach with a multi-criteria fitness function to automate TCP process using GA to increase the code coverage. Li et al. [70] used NSGA-II in which a combination of GPU-based parallel fitness evaluation and parallel crossover computation gave a speed-up of over 100x. Wang et al. [29], [31] used GA on industrial applications like highly configurable systems testing i.e., Cisco Video Conferencing software based on cost-effectiveness measures.

From 2006 onwards, there is a positive growth in the domain of TCP using GA and 2019 shows the maximum number of publications. Also, the most relevant primary studies are obtained in the year 2015, 2016, 2017, 2018 and 2019. 1 primary study (P16) in 2006, 1 primary study (P3) in 2007, 1 primary study (P18) in 2009, 1 primary study (P17) in 2010, 1 primary study (P13) in 2011, 2 primary studies (P1, P19) in 2012, 1 primary study (P20) in 2013, 2 primary studies (P6, P21) in 2014, 4 primary studies (P9, P22, P23, P24) in 2015, 4 primary studies (P5, P26, P27, P35) in 2016, 4 primary studies (P2, P11, P28, P29) in 2017, 5 primary studies (P4, P12, P25, P30, P31) in 2018 and 8 primary studies (P7, P8, P10, P14, P15, P32, P33, P34) in 2019 are conducted. There is no relevant study in the year 2008.

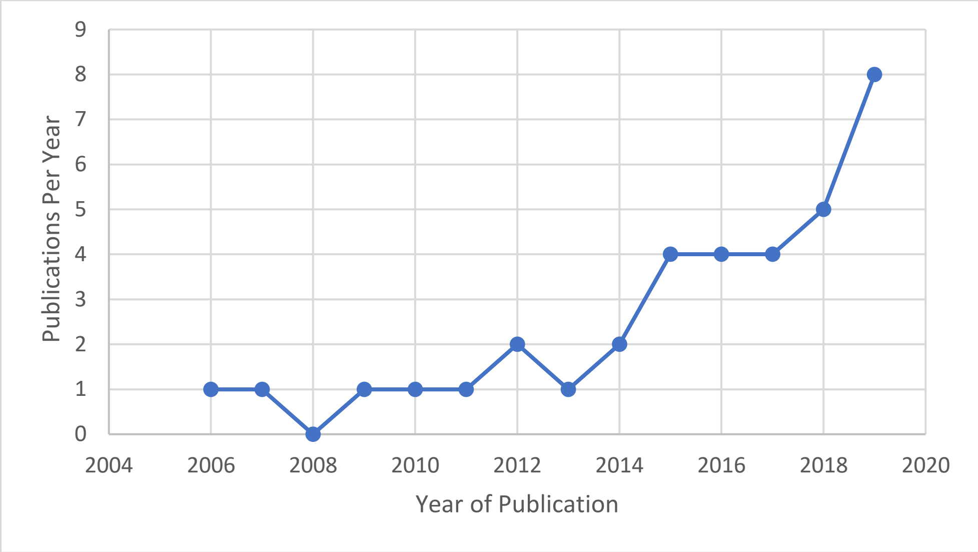

Fig. 4.1 Classification of primary studies regarding publication year

4.2 Publication Trend (RQ 1.2)

This section discusses two domains i.e., type of electronic database and publication used by the researchers as shown in Fig. 4.2. We extracted 35 research papers using four electronic databases i.e., IEEE, Springer, Elsevier and ACM. There are 23 conference papers and 12 journal papers. The majority of publications were in IEEE Transactions on Software Engineering, Journal of Systems and Software, IEEE Transactions on Industrial Informatics, Journal of Systems and Software, International Conference on Software Engineering Companion and so on. Most of the primary studies were presented in Springer and IEEE Xplore, followed by ACM and Elsevier.

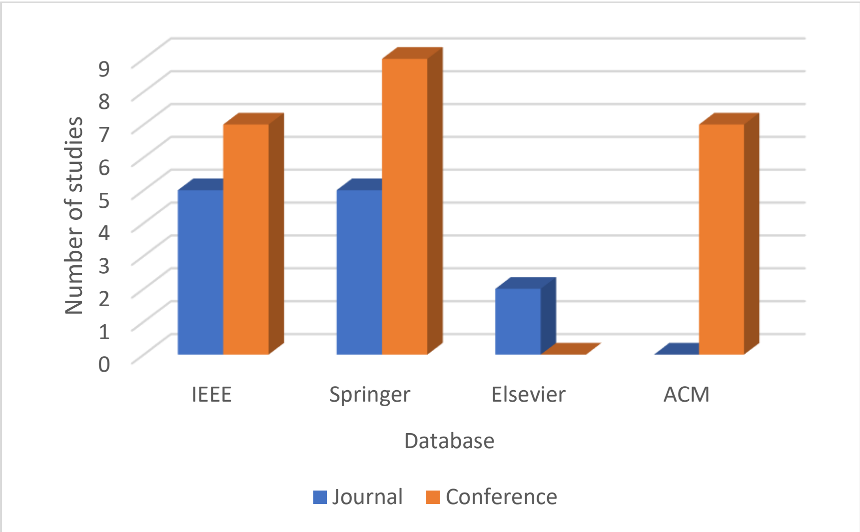

Fig. 4.2 Publication trend of selected studies

4.3 TCP Classification (RQ 1.3)

This segment briefs about TCP techniques used by the studies as represented in Fig. 4.3. We broadly classify these techniques into five categories as coverage-based, fault-based, requirement-based, history-based and combination of various techniques. The mainly used TCP technique is the combination of TCP technique followed by coverage-based technique. History, fault, coverage and requirement-based technique are already discussed in section 1.2. 40% of studies used a combination of TCP techniques that includes: fault + search based technique (P5, P6, P15), coverage + requirement + risk based technique (P8), fault + requirement based technique (P10), fault + coverage based technique (P12, P24, P25, P27, P30, P33, P34, P35) and requirement + coverage based technique (P22).

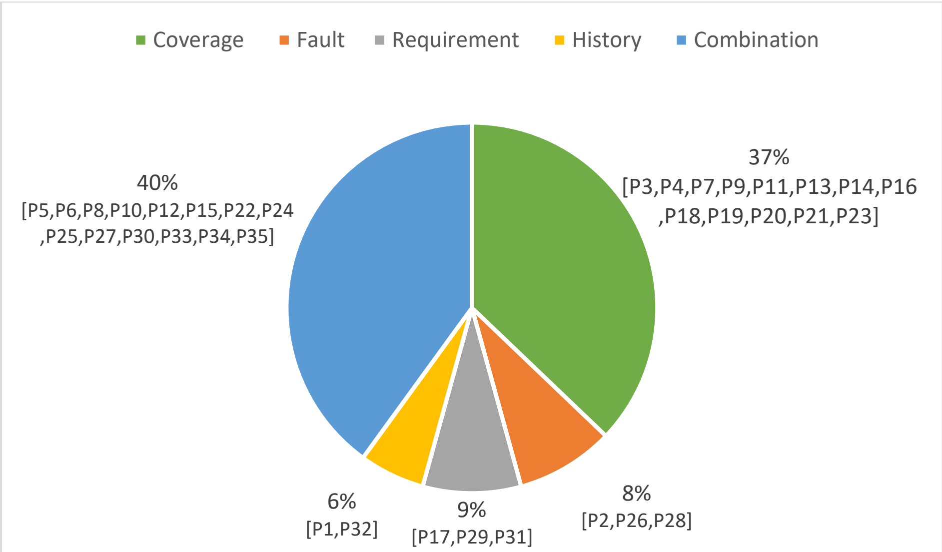

Fig. 4.3 TCP classification

4.4 Genetic Algorithm (RQ 2.1)

This section discusses the types of GA used in TCP by the researchers as shown in Fig. 4.4. We classify four types of GA as simple GA, NSGA-II, two or more GAs and proposed GA. Simple GA is the authentic GA used to arrange the test cases in TCP. There are 12 studies (34%) that used simple GA and it is a primarily used by many researchers as compared to other types of GA.

NSGA-II (Non-dominated Sorting Genetic Algorithm II) is a well-known Multi-Objective Evolutionary Algorithm (MOEAs), the extension of multi-objective optimization. It depends on the Pareto dominance theory to optimize the problems and widely used for prioritizing problems. It is a very popular algorithm used in branches like civil, mechanical, electrical and system engineering other than computer science

[84]. It selects the optimal solutions with regard to the goals among all the solutions and the selection is made by non-dominated sorting. NSGA is classified into three categories, Original NSGA, NSGA-II and NSGA-III. It also adopts elitism to gain fast convergence and crowding distance is used to promote diversity. There are 8 studies (23%) that used NSGA-II. Li et al. [70] presented the GPU-based parallel MOEAs and three novel parallel crossover evaluation schemes in NSGA-II with the 100x speed-up for TCP compared to CPU-based. Marchetto et al. [72] worked with NSGA-II in TCP to increase fault detection, both technical and business relevant. Bian et al. [73] used NSGA-II to prioritize Guava's collection package considering coverage of the changed statements. Parejo et al. [77] used NSGA-II in functional and non-functional properties of Highly-Configurable Systems (HCSs) with seven novel objective functions that provide early fault detection. He also showed superior results with multi-objective prioritization over mono-objective prioritization [77]. Arrieta et al. [79] proposed four objectives for testing industrial Cyber-Physical Systems (CPSs) and designed operators were integrated with NSGA-II that gave an optimal solution in terms of Hypervolume (HV) quality indicator. Pradhan et al. [85] proposed search-based test case prioritization (STIPI) with four objectives using NSGA-II and found reliable performance compared with other techniques using real sets of test cases from Cisco.

Two or more GAs is the combination of more than one GA. There are 4 primary studies (12%) that used two or more GAs to give the optimal solution. Khanna et al. [6] used two GA-based algorithms i.e., WBGA and NSGA-II, in TCP. Wang et al. [29] applied four evolutionary algorithms i.e., RWGA, NSGA-II, MOCell and PAES to resource-aware multi-objective optimization. Mukherjee and Patnaik [58] used GA and MOGA for TCP with reference to the 0/1 Knapsack problem. Arrieta et al. [76] used RWGA and WBGA for configurable CPSs.

Proposed GA means enhanced/modified GA and there are 11 primary studies (31%) that proposed the algorithm. Huang et al. [4] derived a genetic algorithm i.e., GA-hist based on historical records for cost-cognizant TCP that improved the fault detection rate compared to other traditional history-based prioritization techniques. Carballo et al. [28] presented a Biased Random-Key Genetic Algorithm (BRKGA) for TCP problem (TCPP) using a series of real numbers and a decoder. BRKGA outperformed well with large scenarios [28]. Yan et al. [32] proposed an Adaptive Simulated Annealing Genetic Algorithm (ASAGA) which removes the shortcomings of sGA and SAGA. Yuan et al. [40] derived Epistasis in GA with Epistatic Test Case

Segment (ETS) for TCP and it was proven effective and efficient. Yadav and Dutta [53] proposed an algorithm using GA for ordering test cases to achieve maximum statement coverage. Walcott et al. [59] proposed GA as GAPrioritize to reorder the test suites with time constraints. Di Nucci et al. [75] proposed a Hypervolume-based Genetic Algorithm (HGA) using test coverage criteria. Pradhan et al. [78] proposed a Cluster-based Genetic Algorithm with elitist selection (CBGA-ES). Other primary studies also proposed the method for sequencing the test cases in TCP using GA.

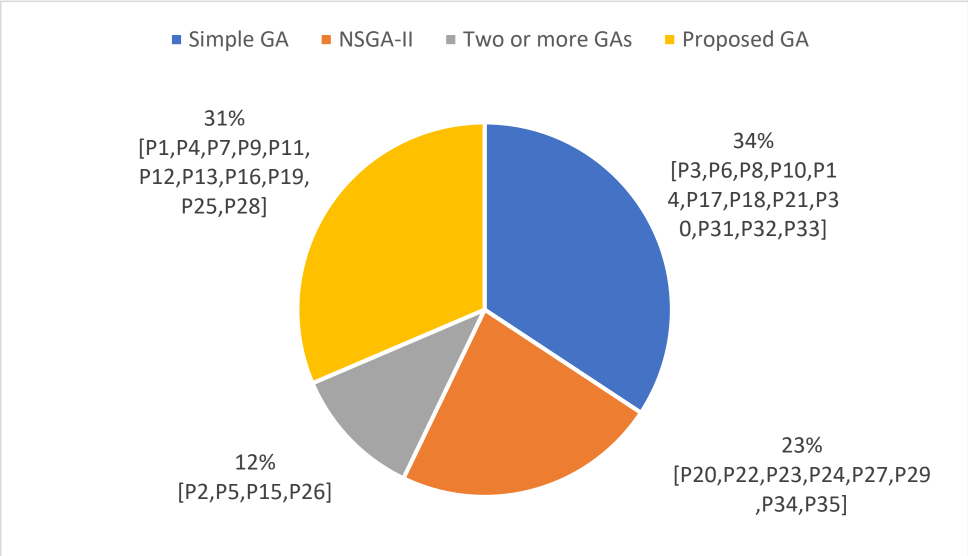

Fig. 4.4 Types of GAs used in TCP

4.5 Encoding Scheme (RQ 2.2)

The initial stage of the GA is the population representation in various forms like binary, integer, character and decimal format. We have classified various encoding schemes as shown in Fig. 4.5. Permutation encoding is helpful for ordering problems where every chromosome is a string of numbers that tells the implementation flow of test cases. It is a widely used encoding scheme. 15 primary studies (41%) used permutation encoding. Binary encoding is a very popular encoding scheme where every chromosome is described as a string of bits i.e., 0 or 1. 6 selected studies (18%) considered binary encoding schemes. Real and permutation encoding is the combination of permutation encoding and value encoding. Value encoding is effective for complex values such as real numbers. There is only 1 study (3%) that used a combination of these two encoding schemes. 12 studies (35%) did not provide the type of encoding scheme used by them. Li et al. [14] have used other type encoding schemes (3%) based on the ordering of chromosomes.

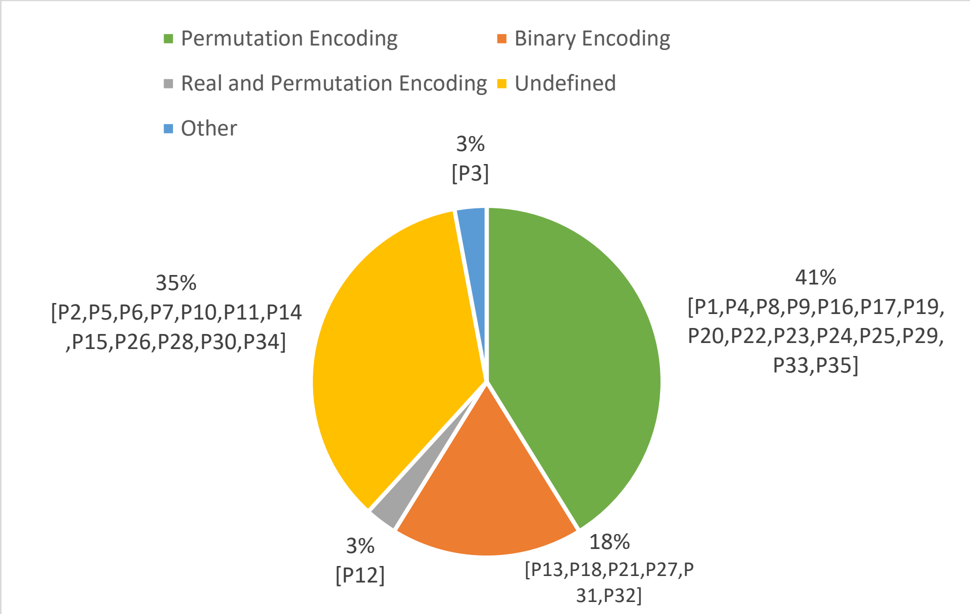

Fig. 4.5 Various encoding schemes used by selected studies

4.6 Population Size (RQ 2.3)

Fig. 4.6 represents the distribution of selected studies based on the population size. We have classified seven categories for population size. 3 studies (8%) have a population size equal to 50, 7 studies (20%) have a population size of 100, 3 studies (9%) have a population size of 200, 2 studies (6%) have a population size of 250, 5 studies (14%) have population size that varies, 10 studies (29%) did not share the population size and 5 studies (14%) have population size as 128, 10, 256, 2*test suite size and 20 for the selected studies [40], [54], [70], [72], [83] respectively.

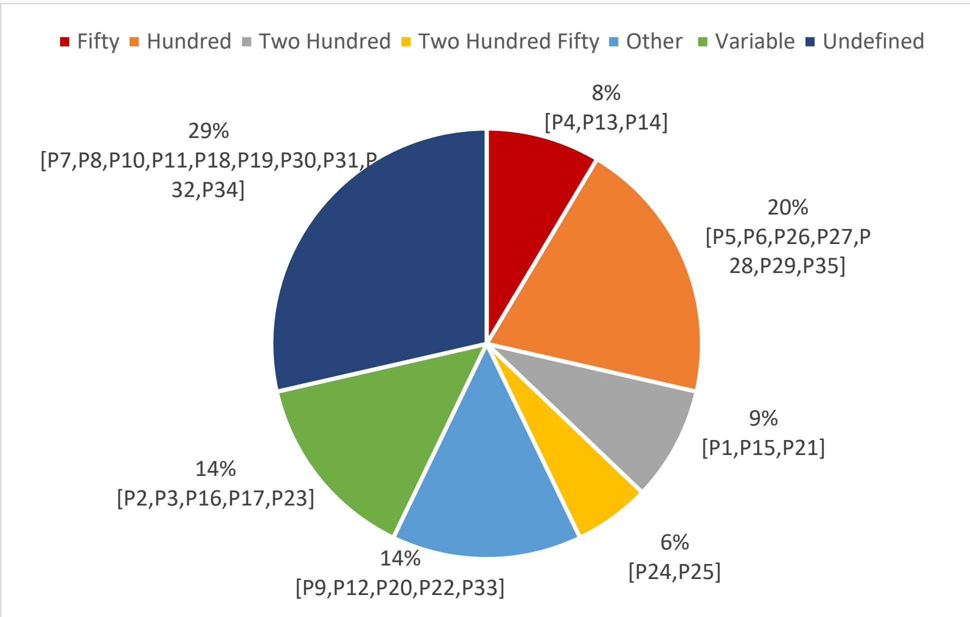

Fig. 4.6 Distribution of studies w.r.t population size

4.7 Generation Size (RQ 2.4)

Generation is another essential measure used in GA to tell the repetitions of the whole process in GA. Fig. 4.7 shows the distribution of generation size with nine categories. 4 studies (11%) used generation size as 50, 2 studies (6%) used generation size as 100, 2 studies (6%) used generation size as 250, 1 study (3%) used generation size as 400, 1 study (3%) used generation size as 500, 3 studies (8%) used generation size as 1000, 6 studies (17%) used generation size that vary, 13 studies (37%) had not defined generation size and 3 studies (9%) used 6*n, 5 and 75 as generation size for the primary studies [28], [54], [58] respectively.

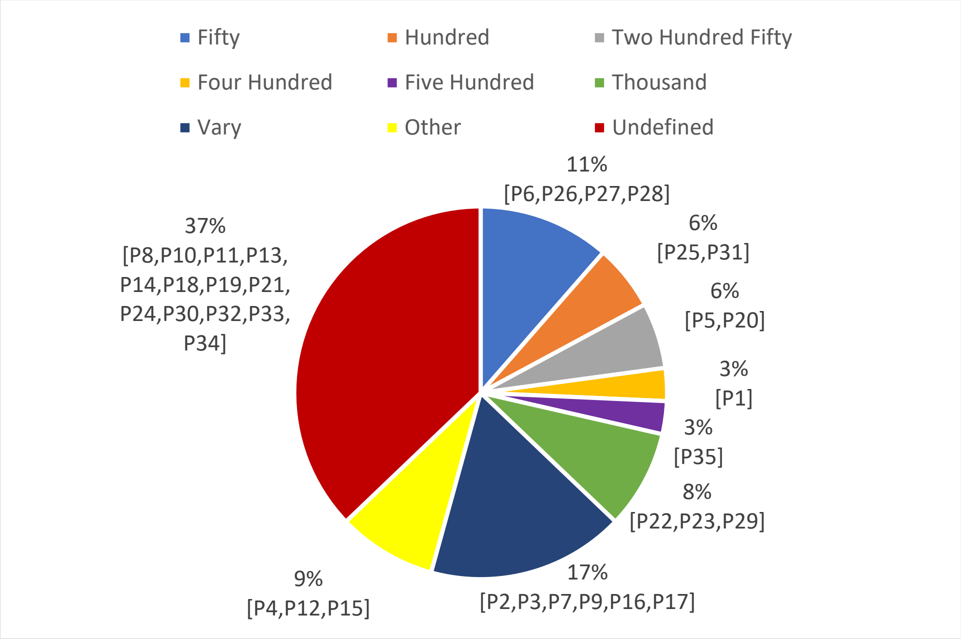

Fig. 4.7 Distribution of studies w.r.t generation size

4.8 Selection Operator (RQ 2.5)

Selection, crossover and mutation operators are three leading operators of GA to give the optimal result. Fig. 4.8 shows the type of selection operator used in GA and we have classified them into five categories. Roulette Wheel selection choose individual based on their fitness. 19 studies (54%) used roulette wheel selection and it is observed to be the highly used selection operator. In Rank-based selection, the individuals are ranked depending open their fitness value and we generally use this operator when the fitness value of the individuals is very close. 4 studies (12%) used a rank-based selection operator. Tournament selection runs some tournaments from randomly selected tournaments and the winner of each tournament is chosen for the next round i.e., crossover. 6 studies (17%) used tournament operators. Only 1 study

(3%) by Conrad et al. [61] used a combination of three selection operators i.e., roulette selection, truncation selection and tournament selection. Truncation selection takes a specified fraction in a population of most fit orderings. 5 studies (14%) did not define the selection operator used in GA for TCP.

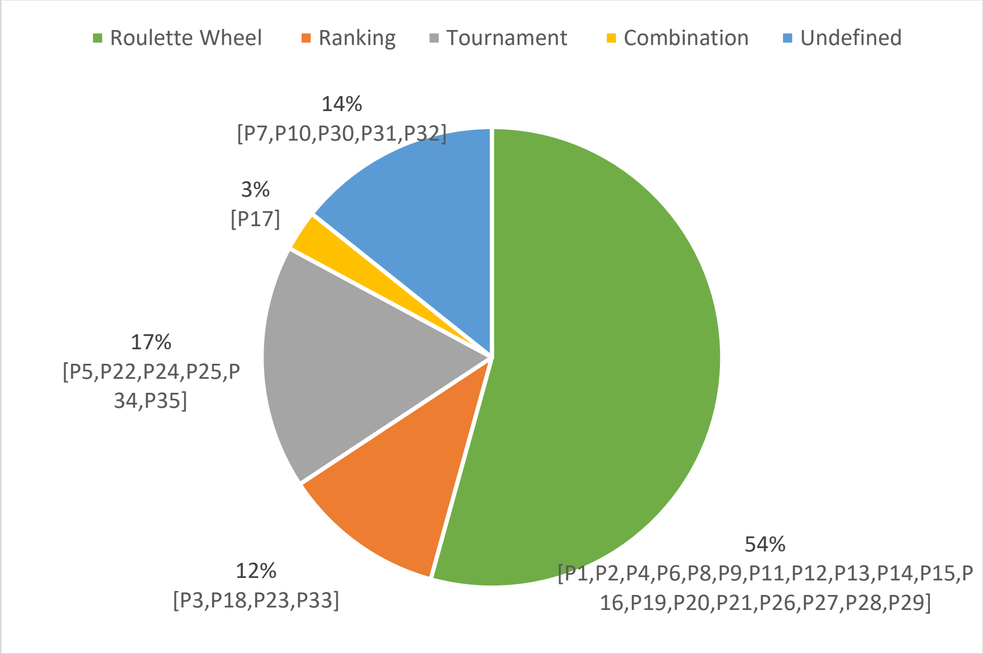

Fig. 4.8 Type of selection schemes

4.9 Crossover Operator (RQ 2.6)

The crossover is performed to produce new offspring by using various crossover operators. We have categorized the crossover operators into seven types as shown in Fig. 4.9 and already discussed these operators in section 1.4. 12 studies (34%) used single point crossover and it is widely used crossover, 2 studies (6%) used ordered crossover, 5 studies (14%) used partially mapped crossover (PMX), only 1 study (3%) used parallel crossover defined by Li et al. [70], 2 studies (6%) used swap crossover defined by Pradhan et al. [78], [85], 4 studies (11%) did not define the type of crossover operator used and 9 studies (26%) used other crossovers i.e., PUX (Parameterized Uniform Crossover), Average crossover, Two-point crossover. PUX crossover operator is used by Carballo et al. [28], Average crossover is used by [38], [54] and two-point crossover is used by [55].

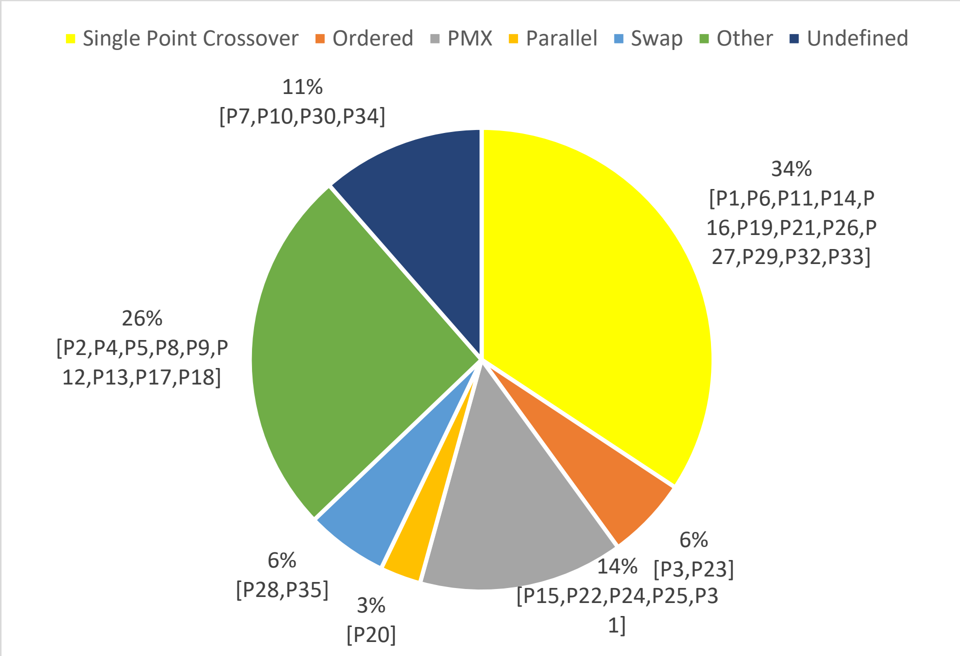

Fig. 4.9 Type of crossover operators

4.10 Mutation Operator (RQ 2.7)

The mutation operator is another genetic operator used after the crossover operator to maintain the diversity to the new generation. We categorize eight types of classification in mutation operator as shown in Fig. 4.10 and already discussed these operators in section 1.4.

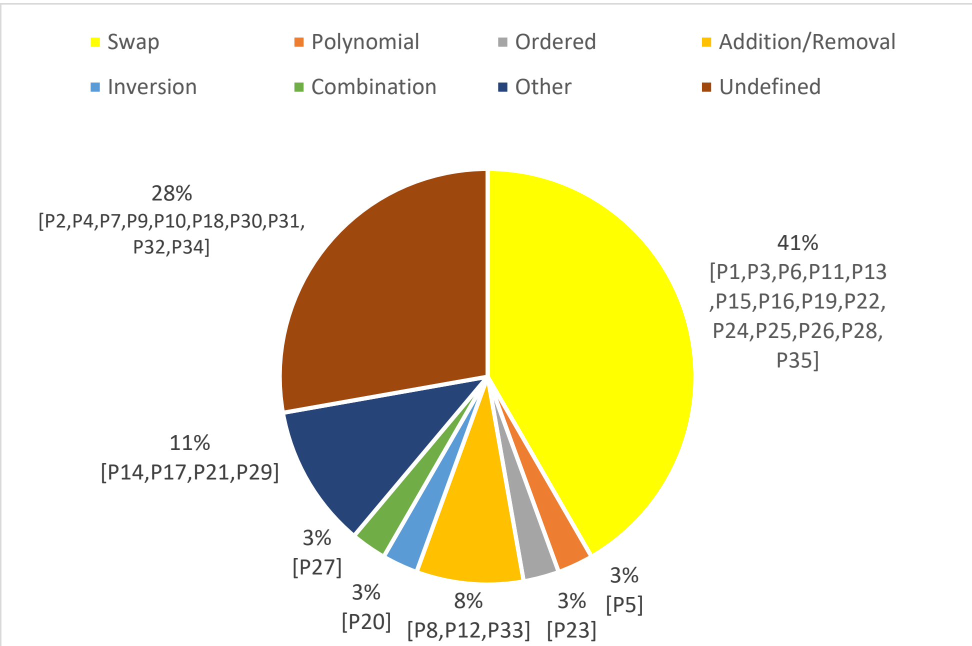

Fig. 4.10 Type of mutation operators

15 studies (41%) used swap mutation, only 1 study (3%) used polynomial mutation defined by Wang et al. [29], only 1 study (3%) used ordered mutation defined by Bian

et al. [73], 3 studies (8%) used addition/removal mutation operator, only 1 study (3%) used inversion mutation, 10 studies (28%) did not define the mutation operator used for finding the optimal solution, 4 studies (11%) used other mutation operator and only 1 study (3%) used a combination of mutation operator i.e., swap + addition/removal + substitution. Substitution mutation selects a random test case and substitutes with another valid test case randomly generated from the test suite [77].

4.11 Crossover Probability (RQ 2.8)

Crossover probability/rate shows the number of times the crossover has occurred for the chromosomes in one generation. We have classified seven crossover rates used by the selected primary studies in Fig. 4.11. There are 2 studies (6%) having crossover rate equal to 1, 8 studies (23%) having crossover rate equal to 0.9, 5 studies (14%) having crossover rate equal to 0.8, 2 studies (6%) having crossover rate equal to 0.7, 3 studies (8%) having crossover rate less than 0.7, 3 studies (9%) having crossover rate varying and 12 studies (34%) having crossover rate as undefined. The maximum selected papers used crossover rate as 0.9 followed by 0.8.

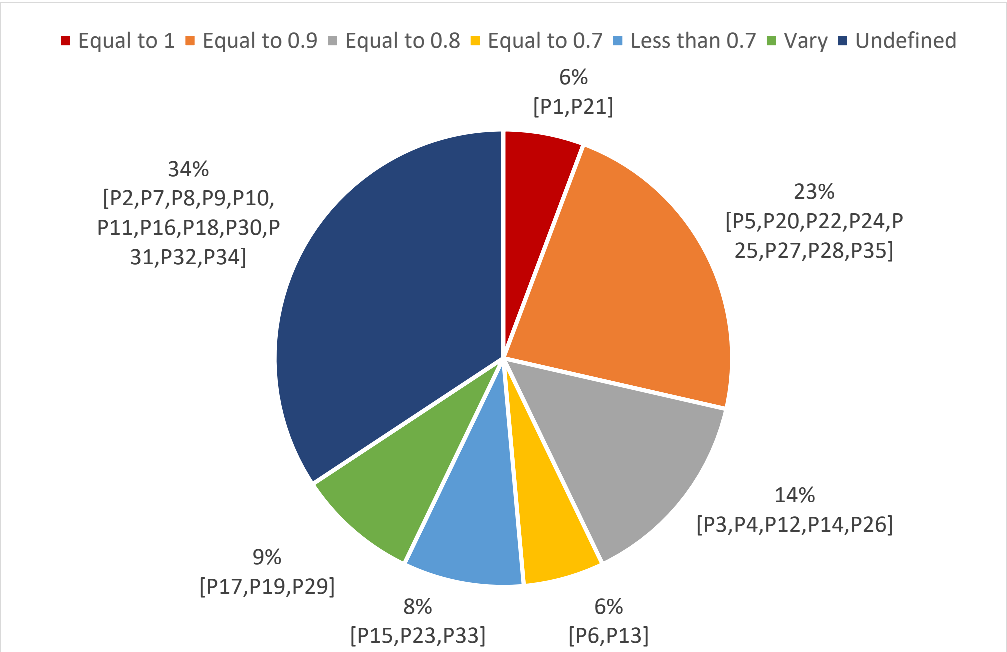

Fig. 4.11 Distribution of studies w.r.t different crossover rates

4.12 Mutation Probability (RQ 2.9)

Mutation probability/rate is also essential for determining the optimal solution. We have classified eight mutation rates used by the selected primary studies as shown in Fig. 4.12. 9 studies (26%) used mutation rate as 1/N, 1 study (3%) used mutation rate equal to 0.4, 12 studies (34%) did not define the mutation rate, 1 study (3%) used mutation rate equal to 0.7, 4 studies (11%) used mutation rate less than 0.1, 1 study (3%) used mutation rate equal to 0.2, 3 studies (9%) used mutation rate equal to 0.1 and 4 studies (11%) used mutation rate that varies. The mainly used mutation rate is 1/N as compared to other mutation rates.

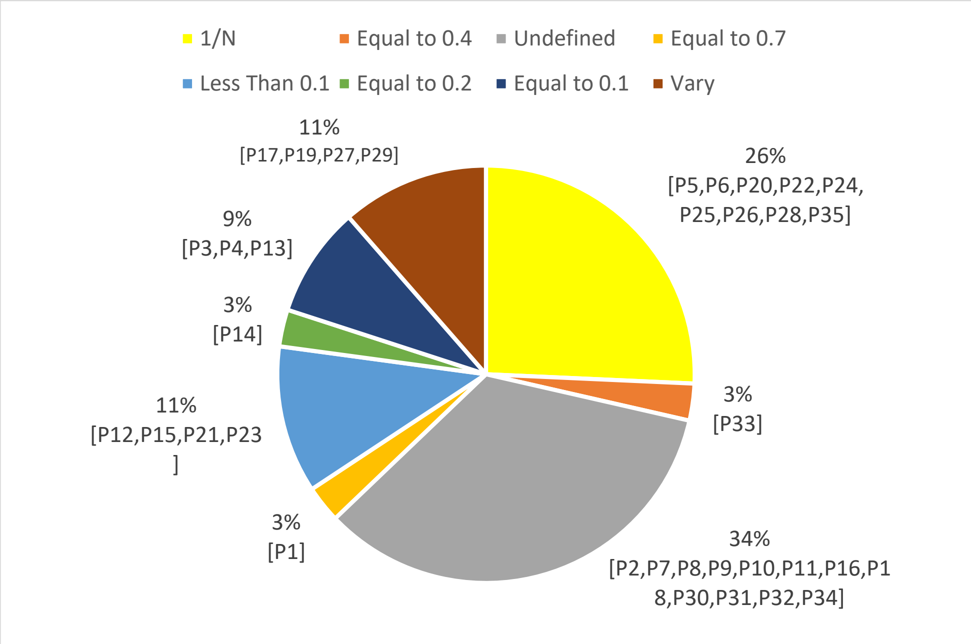

Fig. 4.12 Distribution of studies w.r.t different mutation rates

4.13 Tools (RQ 3.1)

This section discusses the tools used by the primary studies as represented in Table 4.1. Fig. 4.13 represents the percentage of studies that gave the tool information is 35% with 13 primary studies and the percentage of studies that did not share the tool information is 65% with 22 primary studies.

Cantata++[2] is a software testing tool that is used to perform unit and integration testing. SPLOT (Software Product Lines Online Tools) is used for feature models in software product lines. Emma is a set of tools that are used for code coverage. The Linux processing tracking tool calculates peak memory usage and tells the overall system and customer time requirement in TCP. Jester is a JUnit test tester used to find the code executed by running the test cases but not tested in general. MOTCP and MOTCP+ are used to prioritize the test case using GA. Gcov profiling tool and Unix diff tool are used for code coverage information. Valgrind profiling tool is used for memory profiling and memory debugging. Simulink is a graphical tool used for modeling and simulating the systems. SPLCAT tool is used for the generation of the pairwise suite and PLEDGE tool is used for random generation of products [77]. Ant-Java based build tool is meant for the creation and implementation of the test case. Jumble is a mutation testing tool to provide the effectiveness of test cases. Selenium testing tool is required for test case generation and fault seeding. The replay tool is used for application testing, as mentioned by Khanna et al. [84].

TABLE 4.1 Commonly used tools among primary studies

Tools Used	Primary Studies	Number of Primary Studies
Cantata++[2]	P3	1
SPLOT	P7	1
Emma	P2,P16,P30,P34	4
Linux processing tracking tool	P16	1
Jester	P16	1
MOTCP and MOTCP+	P22	1
gcov profiling tool	P24,P25	2
Unix diff tool	P24	1
Valgrind profiling tool	P24	1
Simulink	P26,P29	2
SPLCAT	P27	1
PLEDGE	P27	1
Ant-Java based build tool	P30	1
Jumble	P33	1
Selenium testing tool	P2,P34	2
Replay tool	P34	1

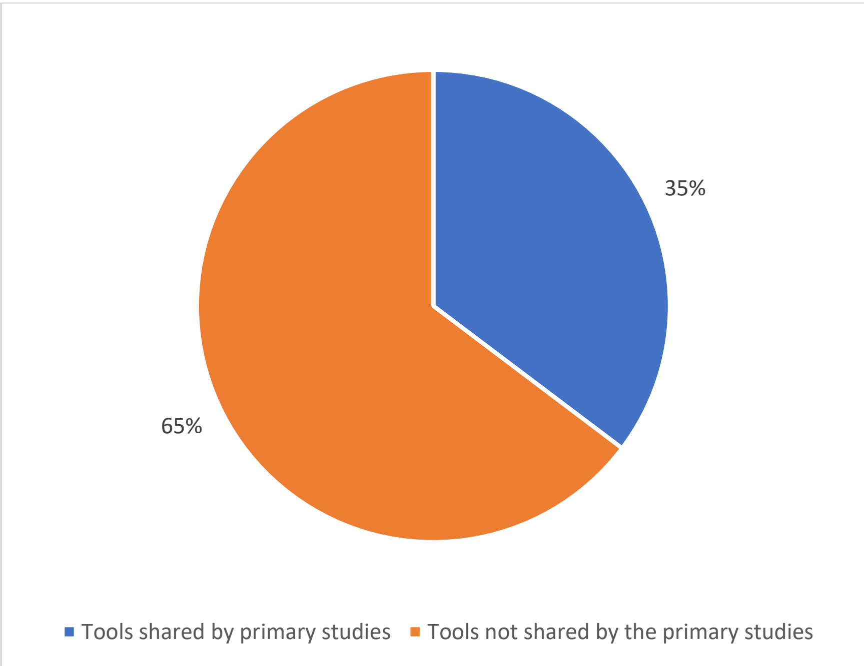

Fig. 4.13 Percentage of tools shared

4.14 Running Environment (RQ 3.2)

Table 4.2 presents the distribution of the running environment i.e., Operating System (OS), Random Access Memory (RAM) and Processor used by selected primary studies. The various types of OS used by studies are classified as: only 1 study used UNIX OS, 6 studies used Windows OS, 2 studies used Linux OS, 2 study used CentOS, which is the upgraded version of CentOS 5 distribution series, only 1 study used Ubuntu OS and only 1 study used MAC OS. The minimum RAM size is 1 GB used by 2 studies and the maximum RAM size is 1024 GB used by only 1 study. 14 studies mentioned the processor used by them, where the most common processors are Intel Core i7, Intel Pentium and Intel Xeon.

TABLE 4.2 Distribution of studies w.r.t running environment

PAPER ID	**OS**	**RAM**	**PROCESSOR**
P1	Unix	N/A	N/A
P2	Windows	N/A	Intel (R) Core(TM) i3-4150 CPU at 3.5GHz
P3	N/A	N/A	N/A
P4	N/A	48 GB	Quad Core Intel Xeon E5530 processor at 2.40GHz
P5	N/A	N/A	N/A
P6	Windows 7	4GB	Intel Core i7 at 2.3GHz
P7	N/A	N/A	N/A
P8	N/A	N/A	N/A
P9	CentOS 6.0 server	16 GB	Intel Xeon (R) E5-2620
P10	N/A	N/A	N/A
P11	N/A	N/A	N/A
P12	N/A	N/A	N/A
P13	N/A	N/A	N/A
P14	N/A	N/A	N/A
P15	N/A	N/A	N/A
P16	Linux	1 GB	Intel Pentium 4 processor at 1.80GHz
P17	N/A	N/A	N/A
P18	N/A	N/A	N/A
P19	N/A	N/A	N/A
P20	Windows 7	4 GB	Intel Core i5-2300 CPU at 2.80GHz
P21	N/A	N/A	N/A
P22	Windows 8	8 GB	Intel Core i7 at 2.20GHz
P23	CentOS 5.11	16 GB	Intel E5426

TABLE 4.2 Distribution of studies w.r.t running environment (continued)

PAPER ID	OS	RAM	PROCESSOR
P24	Linux 6.5	1024 GB	Intel Xeon X7500
P25	N/A	12 GB	Intel Core i7 processor at 2.40GHz
P26	N/A	N/A	N/A
P27	Ubuntu 14.04	16 GB	Intel i7 processor at 3.4GHz
P28	Mac OS X EI Capitan	16 GB	Intel i7 processor at 2.8GHz
P29	N/A	N/A	N/A
P30	N/A	N/A	N/A
P31	N/A	N/A	N/A
P32	N/A	N/A	N/A
P33	Windows 10	1 GB	Intel Pentium processor at 3.2GHz
P34	Windows	N/A	Intel (R) Core(TM) i3-4150 CPU at 3.5GHz
P35	N/A	N/A	N/A

4.15 Programming Language (RQ 3.3)

Fig. 4.14 shows various languages used by the selected studies. 4 studies (12%) used C language, 4 studies (11%) used C++, 8 studies (23%) used Java, 4 studies (11%) used Matlab, only 1 study (3%) used both C and C++, 14 studies (40%) did not specify any language used by them for the implementation. Maximum studies used Java as their programming language.

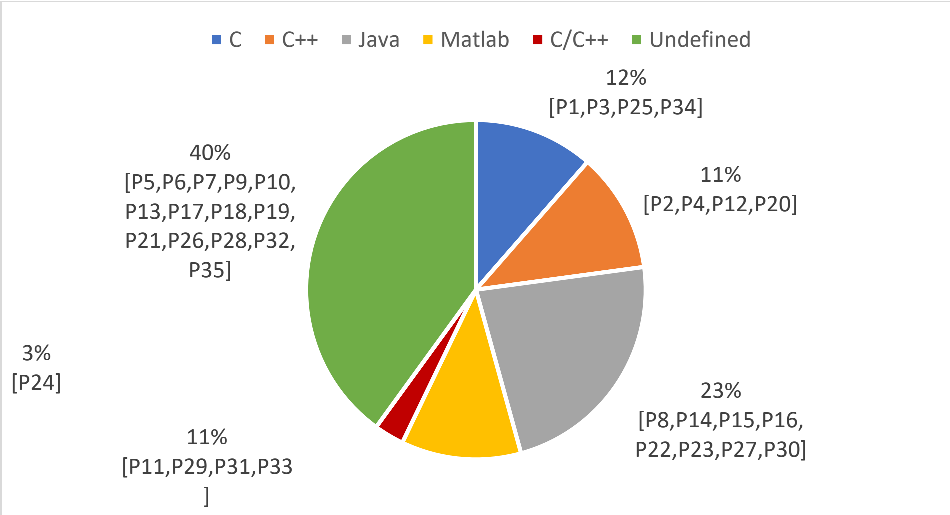

Fig. 4.14 Distribution of studies w.r.t programming language

4.16 Performance Metric (RQ 4.1)

The performance metric is essential for checking the effectiveness of GA used in TCP to find the optimal solution. Fig. 4.15 shows the various types of metrics used by the researchers and we have classified the metrics into five types i.e., APFD, FDC + Others, APEC, APFD + Others and Others. APFD is discussed in section 1.2. Out of 35 selected primary studies, 12 studies (34%) used only APFD metric. Only 2 studies (6%) used APFD + other metric i.e., Yadav and Dutta [56] used APFD + FEP and Kanugo et al. [80] used APFD + APSC.

Fault Detection Capability (FDC) is used to compute the test case effectiveness. It finds the hit rate of the test case and detects faults during test case execution rather than identifying the type of fault [76]. In our study, 5 papers (14%) had used FDC along with other metrics. Wang et al. [29] used FDC + TT + PD + TRU, Wang et al. [31] used FD + PE + FPC + cost measure, Arrieta et al. [76] used FDC + FDT, Pradhan et al. [78] used FDC + CC + APIC + SC, Pradhan et al. [85] used FDC + CC + APIC + SC.

Average Percentage of Element Coverage (APEC) shows element coverage by the ordered test suite i.e., statements/blocks/decisions. It is alike to APDC metric with the change that the evaluation is performed on the coverage of the element rather than fault coverage. Li et al. [14] used APBC + APDC + APSC, Carlallo et al. [28] used APC, Yuan et al. [40] used APSC, Yadav and Dutta [53] used APSC, Mishra et al. [54] used APSC, Jun et al. [55] used APBC, Li et al. [70] used APSC + EET, Bian et al. [73] used APCC + APSC + EET. Out of 35 selected studies, 8 studies (23%) used APEC metric.

Only 8 studies (23%) used other metrics to analyze the effectiveness of GA in TCP. Yan et al. [32] used similarity measure, Mishra et al. [38] used TCW, Azam et al. [41] used CP + EC + CHG + FI + COM + TR, Conrad et al. [61] used CE, Masri et al. [68] used fitness (combination) metric, Ray and Mohapatra [71] used some other measure to check the reliability, Arrieta et al. [79] used HV metric, Kumar and Sujata [81] used TSFD + SM +ASFD metric.

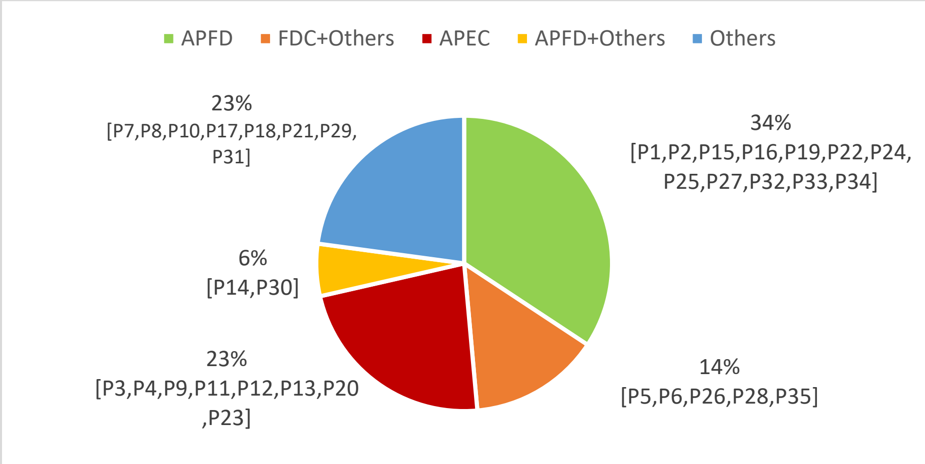

Fig. 4.15 Distribution of studies w.r.t performance metric

4.17 Comparison with other techniques (RQ 4.2)

This section discusses how many researchers validated the results using an existing algorithm or parameters check. It was observed that 28 primary studies compared their proposed technique with the existing algorithm, 4 primary studies compared their proposed study using different parameter settings and only 3 primary studies did not compare the efficacy of the derived algorithm with other existing algorithms or parameters. Fig. 4.16 represents the primary studies that compared their work to check the algorithm's efficiency and shows undefined studies that had not compared their work.

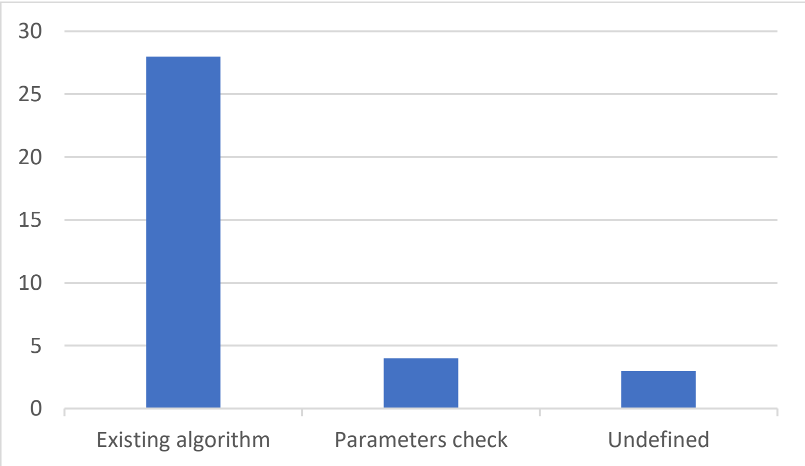

Fig. 4.16 Distribution of studies w.r.t comparison for validation criteria

4.18 Statistical Techniques (RQ 4.3)

The following data describes the statistical techniques used by 16 primary studies that contribute to 46% and 19 primary studies which did not perform the statistical test that contributes to 54% as shown in Fig. 4.17. The types of statistical techniques used by the researchers is shown in Table 4.3.

The Vargha and Delaney test and Mann-Whitney U test are the popular techniques used by most of the studies among the used statistical techniques. 7 papers used both the Vargha and Delaney test and Mann-Whitney U test. In addition to this, 3 papers have also included Bonferroni correction test and 4 papers used Spearman's rank correction coefficient. On the other hand, Friedman's test, Kruskal-Wallis test, Tree model construction, Benjamini-Hochberg correction, PCA (Principal Components Analysis), Mann-Whitney test, Two-way permutation, Welch's t-test and severity detection rate of test are used by only 1 study each.

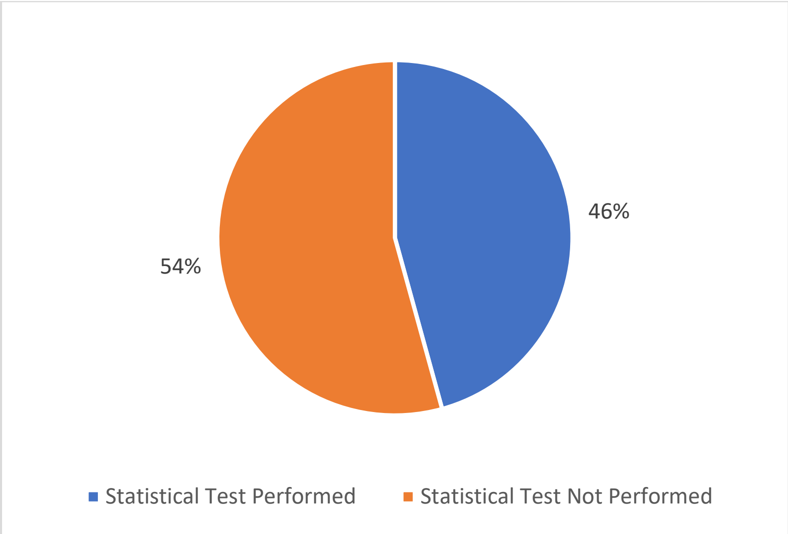

Fig. 4.17 Percentage of statistical test performed

TABLE 4.3 Commonly used statistical tests among primary studies

Statistical Test	Primary Studies	Number of Primary Studies
ANOVA	P3,P15	2
Friedman's test	P4	1
Holm's post-hoc procedure	P4,P27	2
Vargha and Delaney statistics	P5,P6,P23,P24,P25,P27,P28,P29,P35	9
Mann-Whitney U test	P5,P6,P23,P26,P27,P28,P29,P35	8
Kruskal-Wallis test	P5	1
Bonferroni Correction	P5,P23,P24	3

TABLE 4.3 Commonly used statistical tests among primary studies (continued)

Statistical Test	Primary Studies	Number of Primary Studies
Spearman's rank correction coefficient	P5,P6,P22,P26	4
Boxplots	P3,P15,P27	3
Tree model construction	P17	1
Benjamini-Hochberg correction	P22	1
PCA	P22	1
Mann-Whitney test	P22	1
Two-way permutation test	P22	1
Wilcoxon-Signed rank test	P24,P25	2
Welch's t-test	P25	1
Severity detection rate of test	P2	1

4.19 Cost and Execution Time (RQ 5)

Fig. 4.18 shows two parameters i.e., cost and execution time to evaluate the usage of the selected studies. Cost describes if the corresponding study is practical with regard to cost or not. Execution time describes if the strategy results in less execution time. 19 out of 35 studies considered cost factor for effectiveness, whereas 31 out of 35 studies considered execution time i.e., less time is taken for execution.

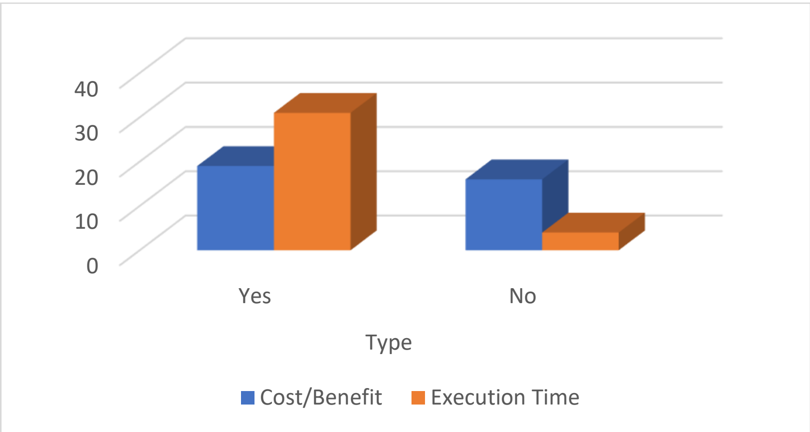

Fig. 4.18 Classification of primary studies w.r.t cost and execution time

4.20 Purpose and Limitations (RQ 6.1)

This section discusses the motive of the primary studies along with their limitations. Table 4.4 shows the purpose of each selected study for using GA in TCP with their respective paper ID and researcher's name.

TABLE 4.4 Goal of primary studies

Paper ID	Purpose
P1	To propose a cost-cognizant TCP by using historical records.
P2	To propose a novel improved multi-objective two-opt algorithm using NSGA-II.
P3	To focus on code coverage TCP techniques i.e., block coverage, decision (branch) coverage and statement coverage.
P4	To propose BRKGA (Biased Random Key GA) in TCP using string of real numbers and a decoder.
P5	To propose resource-aware multi-objective solution using four cost-effectiveness measure.
P6	To introduce search-based multi-objective TCP with cost effectiveness as major objective.
P7	To propose ASAGA (Adaptive Simulated Annealing GA) for local optimization and greater coverage.
P8	To provide statement coverage, requirement priority and execution time using TCP with multi criteria-based GA.
P9	To analyze epistatic in GA for TCP.
P10	To propose intelligent toolkit for automate software quality assurance (SQA) process.
P11	To generate test case for object oriented programming to test the program using statement coverage.
P12	To present code-based technique for providing better statement coverage and fault detection potential.
P13	To design a GA-based TCP based on baseline testing to increase the fault detection rate and reduce the cost.
P14	To design code-based technique using Unified Modeling Language (UML).
P15	To propose two fitness objective functions using GA in 0/1 Knapsack problem w.r.t TCP to maximize the execution of number of modified lines per unit of time and maximizing inheritance edges.
P16	To reorder test suites with time constraints.
P17	To reorder a test suite with wide variety of GA operators and transformation operators.

TABLE 4.4 Goal of primary studies (continued)

Paper ID	Purpose
P18	To propose a technique based on program elements of different types i.e., statements, branches, and def-use pairs.
P19	To propose multi-criteria fitness function that uses flow coverage metrics.
P20	To present a GPU-based parallel fitness evaluation and three parallel computation schemes using NSGA-II.
P21	To propose a prioritization-based technique to achieve high reliability.
P22	To present a new technique with the aim of detecting faults at early stage and efficient in terms of cost.
P23	To present an approach for Guava project by prioritizing its base and collection package w.r.t coverage of changed statements.
P24	To evaluate seven TCP algorithms on a set of five utility programs and introduce an algorithm based on reducing the coverage of data size.
P25	To propose a Hypervolume-based GA (HGA) in TCP with multiple test coverage criteria.
P26	To present an approach for prioritizing the test case based on weight-based search algorithms for configurable CPSs.
P27	To present a real-world case study in Drupal framework based on multi-objective TCP using NSGA-II.
P28	To propose cluster-based GA with elitist selection (CBGA-ES) for reducing the randomness in multi-objective test optimization.
P29	To propose a multi-objective prioritization technique with four objective fitness function and to design different crossover and mutation operator for industrial CPSs.
P30	To present "Fiz-and-reschedule Adaptive" approach with GA in TCP for statement coverage and high fault detection rate.
P31	To present a methodology for prioritizing the test cases by utilizing GA.
P32	To present an approach for TCP using GA based on historical data.

TABLE 4.4 Goal of primary studies (continued)

Paper ID	Purpose
P33	To propose an approach to prioritize the test case using GA and also collect the execution time with different number of generations.
P34	To present three objectives i.e., two with maximization and one with minimization criteria using NSGA-II.
P35	To propose a search-based TCP (STIPI) approach with four objective fitness function using NSGA-II.

Table 4.5 shows the limitations of the primary studies that may lead to ambiguous results under different environments. The data is fetched from threats to validity section and future scope where researchers had discussed various parameters that can be used to enhance the performance of GA in TCP. It is observed that the limitations of many studies are identical i.e., small size programs or very few large size programs, use of minimal case studies, use of specific tools, use of only specific coverage criteria, default parameter settings, same stopping criteria and implementation on small applications. Therefore, it is important to work on these limitations to provide a global optimal solution.

TABLE 4.5 Limitations of the primary studies

S. No.	Limitations	Primary Studies
1	Experiment with either very small size programs or one or two large size programs and few case studies.	P1,P5,P6,P7,P10,P14,P16,P17,P18,P19,P26,P32,P33
2	Experiment with less parameters or techniques.	P2,P10,P34
3	Use of specific tools.	P3,P9,P24
4	Use of only one coverage criteria may give biased outcomes.	P3,P9,P13,P23,P30

TABLE 4.5 Limitations of the primary studies (continued)

S. No.	Limitations	Primary Studies
5	Use of only large scenarios to check the effectiveness.	P4
6	Use of default parameter settings.	P5,P6,P25,P26,P27,P28,P35
7	Use of same stopping criteria.	P5,P6,P26,P28,P29,P35
8	Implementation of the test cases for small software or applications rather than large real-world applications.	P8,P12,P15,P20,P21
9	Use of only one performance metric.	P9,P11,P16,P25
10	Use of only one programming language and testing framework.	P15
11	Use of injected faults rather than real faults.	P22,P26
12	Use of specific server and limited test cases.	P30
13	Algorithm is solved and operated manually.	P31

4.21 Research Discussions and Suggestions (RQ 6.2)

This SLR investigated 35 primary studies out of 522 studies from various electronic databases. The goal is to determine the trend of GA in TCP using a total of 21 questions. It is observed that GA has a great scope in TCP to provide the optimal solution and the area is still open for enhancements. Applying appropriate parameters, operators, databases and tools can provide efficient and reliable results. Based on our analysis, we suggest some recommendations for using GA to prioritize the test cases, which can assist future researchers in properly utilizing GA in TCP.

a) Use of empirical study

It is observed that most of the researchers perform the experiment without comparing it with other futuristic techniques. It is suggested to do an empirical study and experiments to prove the usefulness of the derived technique.

b) Use of public datasets over proprietary datasets

It was observed that the public datasets are more efficacious and structured in terms of reproducing the results, which private data sets cannot do. Future researchers cannot access this data as the studies based on private datasets cannot be repeated. It is recommended to utilize public datasets that can further be helpful to future researchers.

c) Use large real-world applications

The main advantage of large real-world applications is that the problem will get solved by considering more parameters than small applications. Researchers may get optimal results using small applications but may fail to give a globally optimal solution. It is advised to use large real-world applications to get consistent results.

d) Use of prominent evaluation metrics and differentiate the proposed approach with the popular techniques

It is suggested to use metrics for evaluation to validate the outcomes of the derived approach and compare it with other eminent techniques to check its effectiveness. The new techniques should be validated and compared before proven to be effective.

e) Use of more than one performance metric

It is observed that APFD is the most commonly used metric to evaluate the results. Other famous metrics such as APEC, FDC and CE are available to check the efficiency of the proposed technique. Researchers should consider at least two performance metrics to validate the efficiency of the algorithm.

f) Use of more than two coverage criteria

Our study has noticed that code coverage is mainly used as one of the coverage criteria. Generally, researchers use only one coverage criteria in which the result may vary in terms of other parameters. Researchers should use more than one coverage criteria to enhance the effectiveness of the derived algorithm and give more flexibility for prioritizing the test cases using GA.

g) Use of more case studies

It is observed that the researchers use equal or less than two case studies to validate the proposed algorithm and they found different results for dissimilar case studies. It is suggested to consider more than two case studies to check the proposed algorithm on a large scale.

h) Use of standard tools

In our study, we have observed that researchers have used different tools for the implementation. The use of a variety of tools may give bias results. It is recommended to set standard guidelines for using the tools.

i) Define the running environment and language used by the researchers properly

It is noted that only a few researchers have shared the details of the running environment i.e., type of OS, RAM, processor and language used by them to implement the algorithm. It is suggested to give correct information regarding the running environment and language used by the researchers to help future researchers collect the correct data without any ambiguity.

j) Use of statistical techniques

The use of statistical techniques helps to validate the results with more accuracy. The most common statistical techniques are Vargha and Delaney test and Mann-Whitney U test. It is suggested to use statistical techniques for finding out the efficacy of the enhanced algorithm.

CHAPTER-5

THREATS TO VALIDITY

Construct validity can be regarded as constructing research questions and data extraction activity with the data synthesis process. We have used the guidelines of Kitchenham [64] and applied PICOC criteria for minimizing this threat. One author analyzed the selected studies during the data extraction process and another author, a distinguished educator, validated the extracted data by evaluating the selected studies. In case of any discrepancy among the results, a meeting was conducted to arrive at an appropriate result.

Internal validity refers to the generation of the search string. To reduce this threat, we have used advanced search using Boolean operators i.e., ORs and ANDs, in popular databases.

External validity tells the scope by which the results of a systematic review can be achieved in a generalized form. To lessen this threat, we have used popular digital libraries and applied quality assessment followed by inclusion-exclusion criteria without being biased to select maximum relevant studies to the best of our understanding.

CHAPTER-6

CONCLUSION

With the help of this SLR, we have tried to present the research trend of TCP using GA. TCP is the most popular technique used for the maintenance of the software. The use of GA in TCP has been widely explored in recent years by many researchers and find significant results. Due to the popularity of GA in TCP is increasing, we have carried out a SLR to provide a thorough analysis of TCP using GA for evaluating and interpreting the research done so far. We have followed a SLR protocols without being biased. We identified 35 primary studies using search strategy with the help of advanced search, quality assessment, data extraction and data synthesis process. We have formulated several research questions based on research trend, types of TCP techniques, types of GA and its operators, evaluation metrics, running environment, tools, programming language, statistical techniques, goals, limitations and suggestions. In summary, this review paper covers various parameters for GA in TCP to optimize the results and provided suggestions that can help future researchers understand and improve effectiveness for prioritizing the test cases.

REFERENCES

[1] Chin-Yu Huang. Performance Analysis of Software Reliability Growth Models with Testing-Effort and Change-Point. Journal of Systems and Software 2005. https://doi.org/10.1016/j.jss.2004.04.024.

[2] A.G. Malishevsky, G. Rothermel, S. Elbaum. Modeling the Cost-benefits Tradeoffs for Regression Testing Techniques. Proc. - International Conference on Software Maintenance 2002. https://doi.org/10.1109/ICSM.2002.1167767.

[3] Ryan Carlson, Hyunsook Do, Anne Denton. A Clustering Approach to Improving Test Case Prioritization: An Industrial Case Study. International Conference on Software Maintenance (ICSM) 2011. https://doi.org/10.1109/ICSM.2011.6080805.

[4] Yu-Chi Huang, Kuan-Li Peng, Chin-Yu Huang. A History-based Cost-Cognizant Test Case Prioritization Technique in Regression Testing. Journal of Systems and Software 2012. https://doi.org/10.1016/j.jss.2011.09.063.

[5] H.K.N. Leung, L. White. Insights into Regression Testing (Software Testing). Proc. - Conference on Software Maintenance 1989. https://doi.org/10.1109/ICSM.1989.65194.

[6] Munish Khanna, Naresh Chauhan, Dilip Sharma, Abhishek Toofani, Achint Chaudhary. Search for Prioritized Test Cases in Multi-Objective Environment During Web Application Testing. Arabian Journal for Science and Engineering 2018. https://doi.org/10.1007/s13369-017-2830-6.

[7] G. Rothermel, R. H. Untcn, Chengyun Chu, M. J. Harrold. Prioritizing Test Cases for Regression Testing. IEEE Transactions on Software Engineering 2001. https://doi.org/10.1109/32.962562.

[8] Hyunsook Do, Siavash Mirarab, Ladan Tahvildari, Gregg Rothermel. The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments. IEEE Transactions on Software Engineering 2010. https://doi.org/10.1109/TSE.2010.58.

[9] S. Yoo, M. Harman. Regression Testing Minimization, Selection and Prioritization: A Survey. Journal of Software: Testing, Verification and Reliability 2013. https://doi.org/10.1002/stvr.430.

[10] Dennis Jeffrey, Neelam Gupta. Improving Fault Detection Capability by Selectively Retaining Test Cases during Test Suite Reduction. IEEE Transactions on Software Engineering 2007. https://doi.org/10.1109/TSE.2007.18.

[11] Sebastian Elbaum, Praveen Kallakuri, Alexey Malishevsky, Gregg Rothermel, Satya Kanduri. Understanding the Effects of Changes on the Cost-Effectiveness of Regression Testing Techniques. Journal of Software: Testing, Verification and Reliability 2003. https://doi.org/10.1002/stvr.263.

[12] G. Rothermel, R.H. Untch, Chengyun Chu, M.J. Harrold. Test Case Prioritization: An Empirical Study. Proc. - IEEE International Conference on Software Maintenance 1999. https://doi.org/10.1109/ICSM.1999.792604.

[13] S. Elbaum, A. G. Malishevsky, G. Rothermel. Test Case Prioritization: A Family of Empirical Studies. IEEE Transactions on Software Engineering 2002. https://doi.org/10.1109/32.988497.

[14] Zheng Li, Mark Harman, Robert M. Hierons. Search Algorithms for Regression Test Case Prioritization. IEEE Transactions on Software Engineering 2007. https://doi.org/10.1109/TSE.2007.38.

[15] Renee C. Bryce, Sreedevi Sampath, Jan B. Pedersen, Schuyler Manchester. Test Suite Prioritization by Cost-based Combinatorial Interaction Coverage. International Journal of System Assurance Engineering and Management 2011. https://doi.org/10.1007/s13198-011-0067-4.

[16] S. Elbaum, A. Malishevsky, G. Rothermel. Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization. Proc. - International Conference on Software Engineering 2001. https://doi.org/10.1109/ICSE.2001.919106.

[17] S. Elbaum, D. Gable, G. Rothermel. Understanding and Measuring the Sources of Variation in the Prioritization of Regression Test Suites. Proc. - International Software Metrics Symposium 2001. https://doi.org/10.1109/METRIC.2001.915525.

[18] D. Leon, A. Podgurski. A Comparison of Coverage-based and Distribution-based Techniques for Filtering and Prioritizing Test Cases. Proc. - International Symposium on Software Reliability Engineering, ISSRE 2003. https://doi.org/10.1109/ISSRE.2003.1251065.

[19] Gregg Rothermel, Sebastian Elbaum, Alexey G. Malishevsky, Praveen Kallakuri, Xuemei Qiu. On Test Suite Composition and Cost-effective Regression Testing. ACM Transactions on Software Engineering and Methodology 2004. https://doi.org/10.1145/1027092.1027093.

[20] Sebastian Elbaum, Gregg Rothermel, Satya Kanduri, Alexey G. Malishevsky. Selecting a Cost-effective Test Case Prioritization Technique. Software Quality Journal 2004. https://doi.org/10.1023/B:SQJO.0000034708.84524.22.

[21] H. Do, G. Rothermel. A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults. IEEE International Conference on Software Maintenance ICSM 2005. https://doi.org/10.1109/ICSM.2005.9.

[22] Hyunsook Do, Gregg Rothermel, Alex Kinneer. Prioritizing JUnit Test Cases: An Empirical Assessment and Cost-benefits Analysis. Empirical Software Engineering 2006. https://doi.org/10.1007/s10664-006-5965-8.

[23] Daya Gupta, Vishal Gupta. Test Suite Prioritization using Nature Inspired Meta-Heuristic Algorithms. International Conference on Intelligent Systems Design and Applications 2017. https://doi.org/10.1007/978-3-319-53480-0_22.

[24] Anu Bajaj, Om Prakash Sangwan. A Survey on Regression Testing using Nature-Inspired Approaches. International Conference on Computing Communication and Automation ICCCA 2018. https://doi.org/10.1109/CCAA.2018.8777692.

[25] John H. Holland. Adaptation in Natural and Artificial Systems. The MIT Press Cambridge 1976.

[26] David E. Goldberg. Genetic Algorithm in Search, Optimization and Machine Learning. Addison Wesley Publishing Company 1989.

[27] Kenneth Alan De Jong. An analysis of the behavior of a class of genetic adaptive systems. University of Michigan, United States 1975.

[28] Pablo Carballo, Pablo Perera, Santiago Rama, Martin Pedemonte. A Biased Random-Key Genetic Algorithm for Regression Test Case Prioritization. IEEE Latin American Conference on Computational Intelligence LA-CCI 2018. https://doi.org/10.1109/LA-CCI.2018.8625203.

[29] Shuai Wang, Shaukat Ali, Tao Yue, Oyvind Bakkeli, Marius Liaaen. Enhancing Test Case Prioritization in an Industrial Setting with Resource

Awareness and Multi-objective Search. Proc. - International Conference on Software Engineering Companion 2016. https://doi.org/10.1145/2889160.2889240.

[30] Cagatay Catal. On the Application of Genetic Algorithms for Test Case Prioritization: A Systematic Literature Review. Proc. - International Workshop on Evidential Assessment of Software Technologies 2012. https://doi.org/10.1145/2372233.2372238.

[31] Shuai Wang, David Buchmann, Shaukat Ali, Arnaud Gotlieb, Dipesh Pradhan, Marius Liaaen. Multi-objective Test Prioritization in Software Product Line Testing: An Industrial Case Study. Proc. – International Software Product Line Conference 2014. https://doi.org/10.1145/2648511.2648515.

[32] Liu Yan, Wenxin Hu, Longzhe Han. Optimize SPL Test Cases with Adaptive Simulated Annealing Genetic Algorithm. Proc. – ACM Turing Celebration Conference 2019. https://doi.org/10.1145/3321408.3326676.

[33] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, Stephen Linkman. Systematic Literature Reviews in Software Engineering – A Systematic Literature Review. Information and Software Technology 2009. https://doi.org/10.1016/j.infsof.2008.09.009.

[34] Todd L. Graves, Mary Jean Harrold, Jung-Min Kim, Adam Porter, Gregg Rothermel. An Empirical Study of Regression Test Selection Techniques. ACM Transactions on Software Engineering and Methodology 2001. https://doi.org/10.1145/367008.367020.

[35] Emelie Engström, Mats Skoglund, Per Runeson. Empirical Evaluations of Regression Test Selection Techniques: A Systematic Review". Proc. - ACM-IEEE International Symposium on Empirical Software Engineering and Measurement 2008. https://doi.org/10.1145/1414004.1414011.

[36] Yogesh Singh. Software Testing First ed. Cambridge University Press, Cambridge 2012.

[37] Naresh Chauhan. Software Testing Principles and Practices First ed. Oxyford University Press, Oxford 2010.

[38] Deepti Bala Mishra, Rajashree Mishra, Arup Abhinna Acharya, Kedar Nath Das. Test Case Optimization and Prioritization Based on Multi-objective

Genetic Algorithm. Springer Singapore 2018. https://doi.org/10.1007/978-981-13-0761-4_36.

[39] Patipat Konsaard, Lachana Ramingwong. Total Coverage based Regression Test Case Prioritization using Genetic Algorithm. International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology ECTI-CON 2015. https://doi.org/10.1109/ECTICon.2015.7207103.

[40] Fang Yuan, Yi Bian, Zheng Li, Ruilian Zhao. Epistatic Genetic Algorithm for Test Case Prioritization. International Symposium on Search Based Software Engineering 2015. https://doi.org/10.1007/978-3-319-22183-0_8.

[41] Muhammad Azam, Atta-ur-Rahman, Kiran Sultan, Sujata Dash, Sundas Naqeeb Khan, Muhammad Aftab Alam Khan. Automated Test Case Generation and Prioritization using GA and FRBS. International Conference on Advanced Informatics for Computing Research 2018. https://doi.org/10.1007/978-981-13-3140-4_52.

[42] Sebastian Elbaum, Alexey G. Malishevsky, Gregg Rothermel. Prioritizing Test Cases for Regression Testing. Proc. - ACM SIGSOFT International Symposium on Software Testing and Analysis 2000. https://doi.org/10.1145/347324.348910.

[43] Ying-Dar Lin, Chi-Heng Chou, Yuan-Cheng Lai, Tse-Yau Huang, Simon Chung, Jui-Tsun Hung, Frank C. Lin. Test Coverage Optimization for Large Code Problems. Journal of Systems and Software 2012. https://doi.org/10.1016/j.jss.2011.05.021.

[44] Heleno de S. De Campos Junior, Marco Antonio P. Arajo, Jose Maria N. David, Regina Braga, Fernanda Campos, Victor Stroele. Test Case Prioritization: A Systematic Review and Mapping of the Literature. Proc. - Brazilian Symposium on Software Engineering 2017. https://doi.org/10.1145/3131151.3131170.

[45] M. del Carmen De Castro-Cabrera, Antonio Garcia-Dominguez, Inmaculada Medina-Bulo. Trends in Prioritization of Test Cases: 2017-2019. Proc. - ACM Symposium on Applied Computing 2020. https://doi.org/10.1145/3341105.3374036.

[46] Muhammad Khatibsyarbini, Mohd Adham Isa, Dayang N. A. Jawawi, Rooster Tumeng. Test Case Prioritization Approaches in Regression Testing:

A Systematic Literature Review. Information and Software Technolgy 2018. https://doi.org/10.1016/j.infsof.2017.08.014.

[47] Hyunsook Do, G. Rothermel. On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques. IEEE Transactions on Software Engineering 2006. https://doi.org/10.1109/TSE.2006.92.

[48] Y. Fazlalizadeh, A. Khalilian, M. Abdollahi Azgomi, S. Parsa. Prioritizing Test Cases for Resource Constraint Environments using Historical Test Case Performance Data. Proc. - IEEE International Conference on Computer Science and Information Technology 2009. https://doi.org/10.1109/ICCSIT.2009.5234968.

[49] Hyuncheol Park, Hoyeon Ryu, Jongmoon Baik. Historical Value-based approach for Cost-cognizant Test Case Prioritization to Improve the Effectiveness of Regression Testing. Proc. – Second International Conference on Secure System Integration and Reliability Improvement 2008. https://doi.org/10.1109/SSIRI.2008.52.

[50] Jung-Min Kim, Adam Porter. A History-based Test Prioritization Technique for Regression Testing in Resource Constrained Environments. Proc. - International Conference on Software Engineering 2002. https://doi.org/10.1145/581339.581357.

[51] H. Srikanth, L. Williams, J. Osborne. System Test Case Prioritization of New and Regression Test Cases. International Symposium on Empirical Software Engineering 2005. https://doi.org/10.1109/ISESE.2005.1541815.

[52] Hema Srikanth, Laurie Williams. On the Economics of Requirements-based Test Case Prioritization. ACM SIGSOFT Software Engineering Notes 2005. https://doi.org/10.1145/1082983.1083100.

[53] Dharmveer Kumar Yadav, Sandip Dutta. Regression Test Case Prioritization Technique using Genetic Algorithm. Advances in Computational Intelligence 2016. https://doi.org/10.1007/978-981-10-2525-9_13.

[54] Deepti Bala Mishra, Namita Panda, Rajashree Mishra, Arup Abhinna Acharya. Total Fault Exposing Potential based Test Case Prioritization using Genetic Algorithm. International Journal of Information Technology 2019. https://doi.org/10.1007/s41870-018-0117-0.

[55] Wang Jun, Zhuang Yan, Jianyun Chen. Test Case Prioritization Technique based on Genetic Algorithm. International Conference on Internet Computing and Information Services 2011. https://doi.org/10.1109/ICICIS.2011.50.

[56] Dharmveer Kumar Yadav, Sandip Dutta. Regression Test Case Selection and Prioritization for Object Oriented Software. Microsystem Technologies 2020. https://doi.org/10.1007/s00542-019-04679-7.

[57] Priyanka Paygude, Shahank D. Joshi. Use of Evolutionary Algorithm in Regression Test Case Prioritization: A Review. International Conference on Computer Networks, Big data and IoT 2019. https://doi.org/10.1007/978-3-030-24643-3_6.

[58] R. Mukherjee, K. S. Patnaik. Prioritizing JUnit Test Cases Without Coverage Information: An Optimization Heuristics Based Approach. IEEE Access 2019. https://doi.org/10.1109/ACCESS.2019.2922387.

[59] Kristen R. Walcott-Justice, Mary Lou Soffa, Gregory M Kapfhammer, Robert Stewart Roos. TimeAware Test Suite Prioritization. Proc. - International Symposium on Software Testing and Analysis 2006. https://doi.org/10.1145/1146238.1146240.

[60] Y. Singh, A. Kaur, B. Suri, S. Singhal. Systematic Literature Review on Regression Test Prioritization Techniques. Informatica 2012.

[61] Alexander P. Conrad, Robert S. Roos, Gregory M. Kapfhammer. Empirically Studying the Role of Selection Operators During Search-based Test Suite Prioritization. Proc. - Annual Conference on Genetic and Evolutionary Computation 2010. https://doi.org/10.1145/1830483.1830735.

[62] Cagatay Catal, Deepti Mishra. Test Case Prioritization: A Systematic Mapping Study. Software Quality Journal 2013. https://doi.org/10.1007/s11219-012-9181-z.

[63] Anu Bajaj, Om Prakash Sangwan. A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms. IEEE Access 2019. https://doi.org/10.1109/ACCESS.2019.2938260.

[64] S. Keele. Guidelines for Performing Systematic Literature Reviews in Software Engineering Ver. 2.3. EBSE Technical Report, EBSE 2007.

[65] Mark Petticrew, Helen Roberts. Systematic Reviews in the Social Sciences: A Practical Guide. Wiley Blackwell 2005.

[66] Jim Buckley, Toms Mens, Matthias Zenger, Awais Rashid, Gunter Kniesel. Towards a Taxonomy of Software Change. Journal of Software Maintenance Evolution: Research and Practice 2005. https://doi.org/10.1002/smr.319.

[67] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, Mohamed Khalil. Lessons from Applying a Systematic Literature Review Process within the Software Engineering Domain. Journal of Systems and Software 2007. https://doi.org/10.1016/j.jss.2006.07.009.

[68] Wes Masri, Marwa El-Ghali. Test Case Filtering and Prioritization based on Coverage of Combinations of Program Elements. Proc. - International Workshop on Dynamic Analysis 2009. https://doi.org/10.1145/2134243.2134250.

[69] Amr Abdel Fatah Ahmed, Mohamed Shaheen, Essam Kosba. Software Testing Suite Prioritization using Multi-criteria Fitness Function. International Conference on Computer Theory and Applications ICCTA 2012. https://doi.org/10.1109/ICCTA.2012.6523563.

[70] Zheng Li, Yi Bian, Ruilian Zhao, Jun Cheng. A Fine-grained Parallel Multi-objective Test Case Prioritization on GPU. International Symposium on Search Based Software Engineering 2013. https://doi.org/10.1007/978-3-642-39742-4_10.

[71] Mitrabinda Ray, Durga Prasad Mohapatra. Multi-objective Test Prioritization via a Genetic Algorithm. Innovations in Systems and Software Engineering 2014. https://doi.org/10.1007/s11334-014-0234-2.

[72] Alessandro Marchetto, Md. Mahfuzul Islam, Waseem Asghar, Angelo Susi, Giuseppe Scanniello. A Multi-Objective Technique to Prioritize Test Cases. IEEE Transactions on Software Engineering 2016. https://doi.org/10.1109/TSE.2015.2510633.

[73] Yi Bian, Serkan Kirbas, Mark Harman, Yue Jia, Zheng Li. Regression Test Case Prioritisation for Guava. International Symposium on Search Based Software Engineering 2015. https://doi.org/10.1007/978-3-319-22183-0_15.

[74] Michael G. Epitropakis, Shin Yoo, Mark Harman, Edmund K. Burke. Empirical Evaluation of Pareto Efficient Multi-objective Regression Test Case Prioritisation. Proc. - International Symposium on Software Testing and Analysis 2015. https://doi.org/10.1145/2771783.2771788.

[75] Dario Di Nucci, Annibale Panichella, Andy Zaidman, Andrea De Lucia. A Test Case Prioritization Genetic Algorithm Guided by the Hypervolume Indicator. IEEE Transactions on Software Engineering 2020. https://doi.org/10.1109/TSE.2018.2868082.

[76] Aitor Arrieta, Shuai Wang, Goiuria Sagardui, Leire Etxeberria. Test Case Prioritization of Configurable Cyber-physical Systems with Weight-based Search Algorithms. Proc. - Genetic and Evolutionary Computation Conference 2016. https://doi.org/10.1145/2908812.2908871.

[77] Jose A. Parejo, Ana B. Sánchez, Sergio Segura, Antonio Ruiz-Cortés, Roberto E. Lopez-Herrejon, Alexander Egyed. Multi-objective Test Case Prioritization in Highly Configurable Systems: A Case Study. Journal of Systems and Software 2016. https://doi.org/10.1016/j.jss.2016.09.045.

[78] Dipesh Pradhan, Shuai Wang, Shaukat Ali, Tao Yue, Marius Liaaen. CBGA-ES: A Cluster-Based Genetic Algorithm with Elitist Selection for Supporting Multi-Objective Test Optimization. IEEE International Conference on Software Testing, Verification and Validation ICST 2017. https://doi.org/10.1109/ICST.2017.40.

[79] Aitor Arrieta, Shuai Wang, Urtzi Markiegi, Goiuria Sagardui, Leire Etxeberria. Employing Multi-Objective Search to Enhance Reactive Test Case Generation and Prioritization for Testing Industrial Cyber-Physical Systems. IEEE Transactions on Industrial Informatics 2018. https://doi.org/10.1109/TII.2017.2788019.

[80] Garima Kanugo, Sukhvir Kaur, Vikas Khullar. Enhanced Test Case Prioritization Technique Based on Fix-and-Reschedule Adaptive Approach and Genetic Algorithm. International Conference for Convergence in Technology I2CT 2018. https://doi.org/10.1109/I2CT42659.2018.9057830.

[81] Kuldeep Kumar, Sujata. Regression Testing using User Requirements based on Genetic Algorithms. Innovations in Computational Intelligence 2017. https://doi.org/10.1007/978-981-10-4555-4_18.

[82] Pushkar Padmnav, Gaurav Pahwa, Dinesh Singh, Saurabh Bansal. Test Case Prioritization based on Historical Failure Patterns using ABC and GA. International Conference on Cloud Computing, Data Science & Engineering 2019. https://doi.org/10.1109/CONFLUENCE.2019.8776936.

[83] Getachew Mekuria Habtemariam, Sudhir Kumar Mohapatra. A Genetic Algorithm-Based Approach for Test Case Prioritization. International Conference on Information and Communication Technology for Development for Africa 2019. https://doi.org/10.1007/978-3-030-26630-1_3.

[84] Munish Khanna, Achint Chaudhary, Abhishek Toofani, Anil Pawar. Performance Comparison of Multi-objective Algorithms for Test Case Prioritization During Web Application Testing. Arabian Journal for Science and Engineering 2019. https://doi.org/10.1007/s13369-019-03817-7.

[85] Dipesh Pradhan, Shuai Wang, Shaukat Ali, Tao Yue, Marius Liaaen. STIPI: Using Search to Prioritize Test Cases Based on Multi-objectives Derived from Industrial Practice. IFIP International Conference on Testing Software and Systems 2016. https://doi.org/10.1007/978-3-319-47443-4_11.