International Journal of Mathematical, Engineering and Management Sciences
Vol. 8, No. 2, 293-315, 2023
https://doi.org/10.33889/IJMEMS.2023.8.2.018

Software Test Case Generation Tools and Techniques: A Review

Abhishek Singh Verma
Department of Computer Science & Engineering,
Dr. A. P. J. Abdul Kalam Technical University, Lucknow, U.P., India.
SET, Sharda University, Greater Noida, U.P., India.
E-mail: abhiverma2005@gmail.com

Ankur Choudhary
Department of Computer Science and Engineering,
SET, Sharda University, Greater Noida, U.P., India.
Corresponding author: ankur.tomer@gmail.com, ankur.choudhary@sharda.ac.in

Shailesh Tiwari
Department of Computer Science and Engineering,
Krishna Engineering College, Ghaziabad, U.P., India.
E-mail: shail.tiwari@yahoo.com

(Received on November 07, 2022; Accepted on December 12, 2022)

Abstract
Software Industry is evolving at a very fast pace since last two decades. Many software developments, testing and test case generation approaches have evolved in last two decades to deliver quality products and services. Testing plays a vital role to ensure the quality and reliability of software products. In this paper authors attempted to conduct a systematic study of testing tools and techniques. Six most popular e-resources called IEEE, Springer, Association for Computing Machinery (ACM), Elsevier, Wiley and Google Scholar to download 738 manuscripts out of which 125 were selected to conduct the study. Out of 125 manuscripts selected, a good number approx. 79% are from reputed journals and around 21% are from good conference of repute. Testing tools discussed in this paper have broadly been divided into five different categories: open source, academic and research, commercial, academic and open source, and commercial & open source. The paper also discusses several benchmarked datasets viz. Evosuite 10, SF100 Corpus, Defects4J repository, Neo4j, JSON, Mocha JS, and Node JS to name a few. Aim of this paper is to make the researchers aware of the various test case generation tools and techniques introduced in the last 11 years with their salient features.

Keywords- Software testing, Test case generation, Test data, Software reliability, Software engineering.

1. Introduction

Software industry is very dynamic, uncertain and volatile, since 19th century to till date new software trends are taking place to meet the growing needs of customers. But throughout this tenure, the quality and reliability of software have always been a prime focus. The increasing dependency of society on the daily use of the software is making software reliability a more important aspect. The reliability of software mainly depends on software testing. Better software testing may lead to more reliable software.

Software Testing is a process of applying existing or new test cases on software for verification during development and for validation of functionality before release. Testing time is an essential key factor that depends on the scope and complexity of the software. It is identified that testing consumes more than 50% software development time as it involves various activities. Testing activities include the following tasks: generation of test cases, forming test suites, execution of test suites, analyze the results of the execution and prepare the report of testing activity (Baresi and Pezzè, 2006; Setiani et al., 2019).

The main of test case generation is to make test suites for detecting bugs in the software. These test cases and test suites will ensure that the system satisfies the reliability, standards, and customer requirements of the software. The test case generation approach contains four main activities: information artifacts, generation mechanism, test case validity and formation of test oracle. These activities are used to represent the basic test case generation process shown in Figure 1 (Clark et al., 2021).

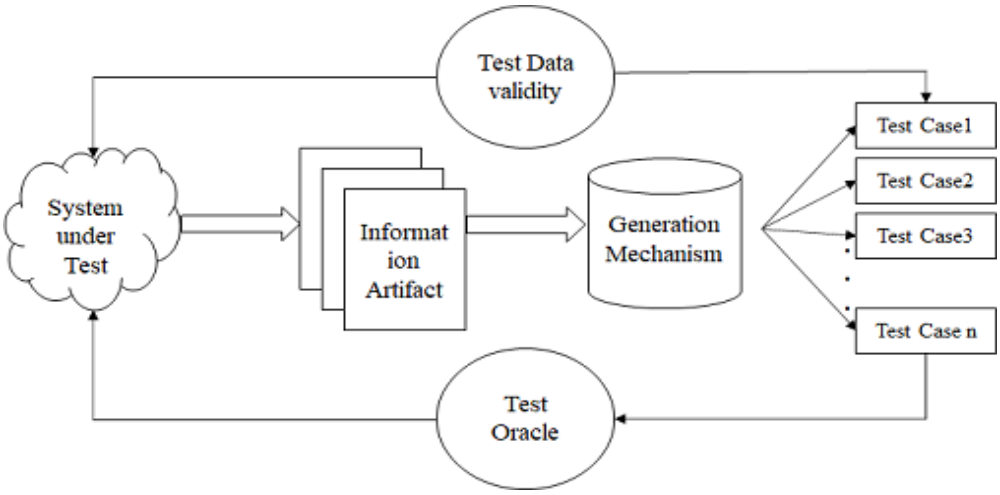

Figure 1. Basic test case generation process.

In practicality, a test case generator is a software program that takes a source code of software to be tested, test case validation criteria, test case specifications and basic data structure definitions of test cases as input parameters and generates test cases as output. Test case generation algorithms use heuristics or some explicit strategy to generate test cases which maximize code coverage or fault coverage capabilities of test cases. According to literature , generating accurate test cases which can fulfil the test requirements is not an easy task (Narciso et al., 2014). In last decade so many researches have been done in the area of test case generation. So, it became necessary to review these important literatures for better classification of approaches and identification of research gaps.

This paper provides a systematic review to present a broad overview of the existing literature on test case generation tools and techniques since 2011. The motivation behind the paper is to focus on evidences for identification of research gap in test case generation tools and techniques.

To perform the review process, authors have included almost all the important literature and presented them in a well-organized manner. This review has been conducted on a set of 125 papers published in conference and journals of repute between year 2011 and 2021. The selection of these papers is done using multistage selection criteria in the field of test suite generation.

Followed by introduction, the structure of this paper is as follows: Section 2, describes the research methods for systematic mapping of existing work which includes the process of review, research questions, search criteria, the inclusion and exclusion criteria and selection procedure for research articles. Section 3 discussed the results and provide the answers of research questions with proper evidences. At last, Section 4 presented the conclusions of the systematic review as well as identifies the future scope for the current study.

2. Research Methods for Systematic Mapping of Literature Review

In last decade, researchers have contributed a lot on test case generation to make the generation process more efficient and truly automated. Systematic review is the way to look back to the studies. It is a rigorous and consistent practice for identifying and classifying related research as much as possible and also focus on selection of relevant findings, analyse data, and perform evaluation or assessment (de Almeida Biolchini et al., 2007). There are some slandered guidelines to conduct systematic review, such guidelines provide the steps to perform systematic review. The authors have followed these guidelines conduct an effective systematic literature review (Budgen and Brereton, 2006; Brereton et al., 2007; Kitchenham et al., 2009).

This paper reviews existing state of the art tools and techniques of test case generation. In this section, authors have summarized the research methods used to perform systematic review covers the review process, research questions, review search and selection criteria. The objective is to mention the highlighted points of existing literature and summarize the available research work in test case generation domain.

2.1 Review Process

As suggested by Brereton et al. (2007), the review process has been conducted in three phases: Planning, Conducting and Reporting.

In the planning phase following activities has been performed:
(i) Analyze the need of literature review.
(ii) Develop a variety of the research questions to address the need.
(iii) Designing the objectives of the review process.

In conducting phase following activities has been performed:
(i) Designing search criteria
(ii) Define review data selection criteria,
(iii) Review selection procedure including inclusion & exclusion selection criteria and selection of the data from existing literature.

Finally, in reporting phase following activities has been performed:
(i) Reports have been prepared which includes findings of results with discussion in various forms such as tables, graphs & descriptions.

The objective of overall review process is to answers the research questions with evidences, identify research gaps and achieve the objectives which may help in future research work (Petersen et al., 2007).

While performing this literature review, following guidelines have been followed (Budgen and Brereton, 2006; Kitchenham et al., 2009). This process involved following steps:

- Selection of different e-databases from where the primary research papers were collected.
- Design a search string to capture all related publications.
- Defining inclusion & exclusion criteria for selection of related publications.
- Designing quality research questions to assess the quality of related papers to fulfil the objective of the review.

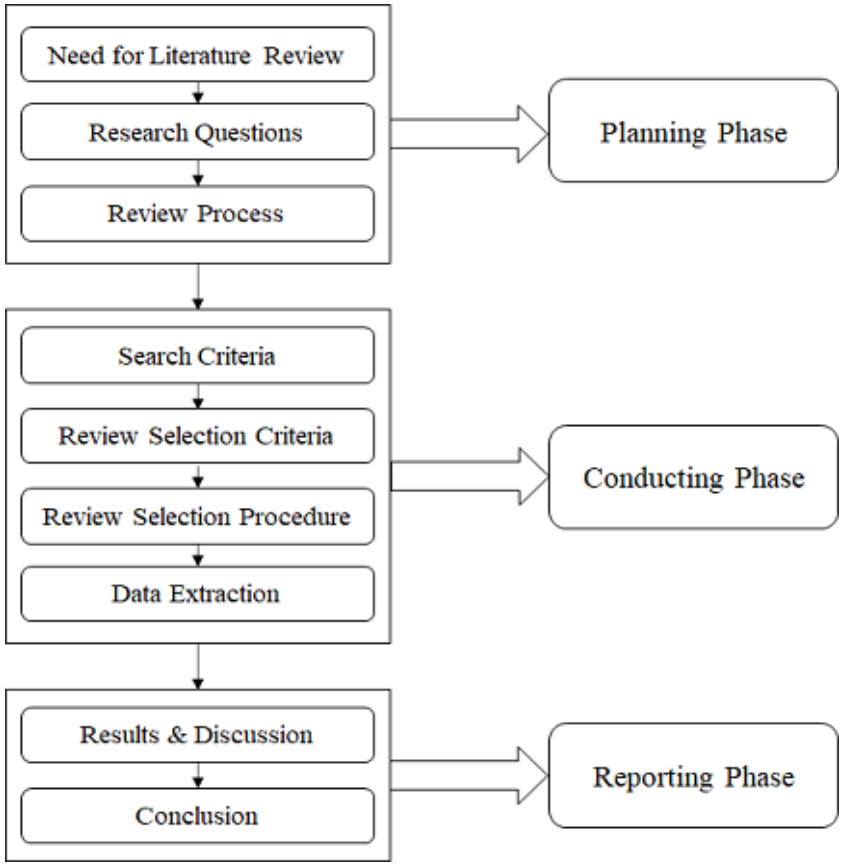

Figure 2. Critical literature review process.

2.2 Research Questions

Test case generation process is suffering from various challenges. To identify and analyze the impact of such challenges and their solutions, various research questions (RQ) have been developed. Six questions have been formed for analysis and synthesis purpose. These questions were formed to analyze the research gaps and identify the main aspects in the development and design of test case generation research domain.

The authors have framed following research questions to analyze the literature and conclude to concrete outcome. The lists of framed research questions for this study are mentioned below:

RQ1: How Test Case Generation techniques are different from each other?
RQ2: Are there any standard programs or datasets available to perform basic test case generation research?
RQ3: What is the ratio of publishing research articles in conferences and journals of repute on test case generation?
RQ4: Which type of testing has been explored a lot for research of test case generation?
RQ5: What are the various tools developed and utilized for automatic test case generation?
RQ6: What are the various optimization algorithms used for optimizing the results of test case generation process?

2.3 Review Search Criteria

The existing literature have been collected by performing different search techniques on various digital libraries such as IEEE, Springer, ACM, Elsevier etc. & different online platforms of DBLP, Google Scholar etc. Table 1shows the sources used in this review for selecting various journal and conference papers.

Table 1. Selected digital libraries for papers.

Digital Library	URL
Google Scholar	https://scholar.google.com/
IEEE	https://ieeexplore.ieee.org/
Springer	https://link.springer.com/
ACM	https://dl.acm.org/
Elsevier	https://www.elsevier.com/
Wiley	https://www.wiley.com/

During searching process, related search keywords have been selected to prepare the search strings. An accurate and systematic approach for searching has been developed. The following steps have been comprised to select publications using search keywords.

- The literature of test case generation has been explored and most repeated keywords are selected for searching more articles.
- Explore matching string, alternative keywords and similar words that frequently used.
- Design search strings by using AND, OR and NOT operators and apply in search engines.
- At last, manually verify the searched publications that the research papers are related to the research questions formed for literature review.

The titles of few of the searched papers were not relevant to this study, so these papers have been discarded. In second step, the abstract & conclusion of the selected papers has been analyzed, if the paper is found reverent to the study, then include these papers otherwise discard it. The related search strings and alternate keywords used for this study are shown in Table 2.

Table 2. Related search string for automatic search.

S. No.	Search String	Alternate Keywords
I	Test Suite Generation	(Test suite AND generation) OR (generate AND test suites)
II	Test Case Generation	(Test case AND generation) OR (generate AND test cases)
III	Test Data Generation	(Test data AND generation) OR (generated AND test data)
IV	Automatic Test Case Generation	(Automatic AND test case generation) OR (generated AND test cases AND automatically)

After completed the exploration process, authors have included the relevant papers published during a period of 11 years starting from year 2011 to year 2021 in the domain of Test Case Generation. Figure 3 shows the no. of publications year by year in last decade.

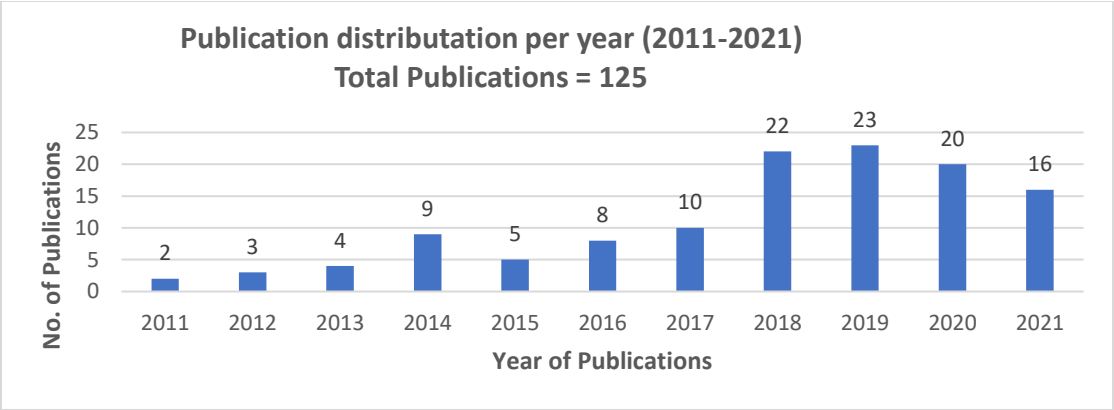

Figure 3. Representation of related papers published year wise in test case generation.

2.4 Selection Criteria of Primary Studies

The literature collection is performed into two. The first phase includes publication title including abstract and conclusion, year & sources, summary of the research study. In second phase, the technical information collected to provide the satisfactory answer to the research questions formed for this systematic literature review.

The screening processes of selection of publications for primary studies are shown in Figure 4 and Figure 5.

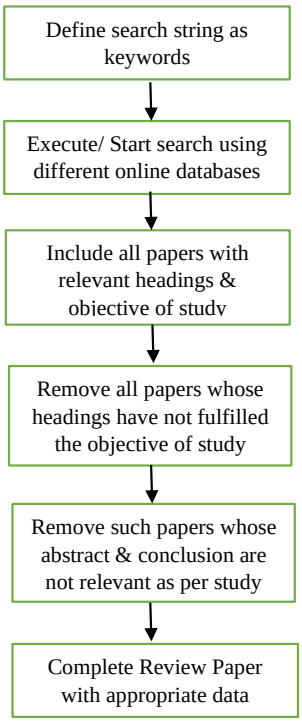

Figure 4. Representation of selection criteria.

After each extraction, the no. of papers is filtered as per extraction selection criteria i.e., remove duplicate papers, title or abstract does not match, paper is not lying-in specified range, paper is either book, thesis, project report or non-peer review publication.

For selection of research papers, the inclusion & exclusion selection criteria are applied to exploit the search process as represented in Table 3. After applying these criteria, some papers were removed from the pool as these papers were not satisfying the search conditions of review process.

Table 3. Inclusion & exclusion selection criteria.

Selection Criteria	ID's	Description
Inclusion Criteria	IC1	Studies that include the topics of test case generation, test suite generation etc. that proposing some approach/ method and applied in some case studies.
	IC2	Papers published in either journals or conference proceedings.
	IC3	Papers including surveys and qualitative work that address the issue of Test Case Generation.
	IC4	Consider the studies in academic & industry.
Exclusion Criteria	EC1	Papers not written in English language.
	EC2	Papers not fulfilling the search condition of keywords.
	EC3	Books, reports, thesis, tutorials and non-peer review publications.
	EC4	Duplicate papers from different resources.
	EC5	Papers are not lying in the range of years of selection criteria.

During searching process, the authors found 738 papers with matching keywords from various digital libraries. After performing 3 round of exclusion process authors have selected 125 papers, which fulfills the search conditions with their titles, abstracts and conclusions. Now, finally 125 papers found suitable for critical review process.

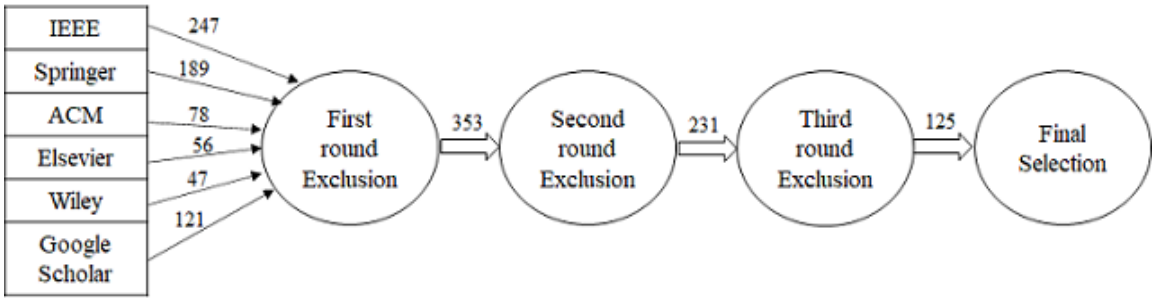

Figure 5. Selection procedure of primary studies.

3. Results and Discussion

As software testing having various activities, generation of test cases is one of the crucial and important activity, as it has a robust impact on the efficiency and effectiveness of the testing process (Zhu et al., 1997; Bertolino, 2007) Test Case Generation is an activity in which test cases are generated either manually or automatically by using any automatic test case generation tool (Prasanna et al., 2011; Avdeenko and Serdyukov, 2021).

This section provides the answer of the research questions formed in this study based on the papers collected in the previous section. To answer of research questions, the relevant evidences of existing literature has been used, which will help the researchers of the domain of test case generation.

Answer to RQ1

To answer the research question 1, existing literature has been explored. It became clear that there exists different type of test case generation techniques such as model based, random based, search-based, specification based and metamorphic software testing approach. Figure 6 and Table 4 further differentiate the various approaches of test case generation and discuss their advantages & disadvantages.

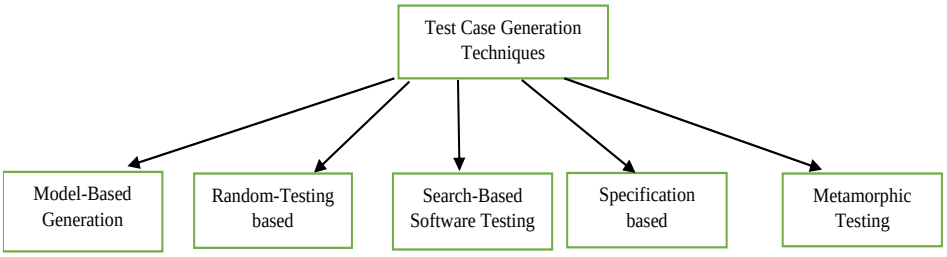

Figure 6. Test case generation techniques.

Table 4. Descriptions of test case generation techniques.

Name of Technique	Description of Technique	Advantages	Disadvantages	References of Technique
Model-Based Generation	Model-based testing is a technique in which the test cases are generated automatically from application models written in EFSM (Extend Finite-State Machine). Such models consist of business logic & is just a few lines of code.	(1) Communication improvements. (2) More Cost-Effective.	(1) Increasing Complexity (2) Adaptation To the New Approach	(Păsăreanu et al., 2009; Nguyen et al., 2012; Enoiu et al., 2013; Shirole and Kumar, 2013; Nabuco ans Paiva, 2014; Li et al., 2018; Fellner et al., 2019)
Random-Testing based	Random-testing method involves creating random and independent inputs to test programmes. It's a sort of black-box testing. The output created is compared to the software specifications to see whether the outcome is correct or not.	(1) Do not require to know programming languages. (2) it is cheaper, simpler and easier.	(1) Many of the tests are redundant and unrealistic. (2) More time is spent on analysing results.	(Liu et al., 2010; Zhou, 2010; Huang et al., 2012; Ramler et al., 2012; Bala Mishra et al., 2017; Wetzlmaier and Ramler, 2017; Lemberger, 2020)
Search-Based Software Testing	The process of utilizing search techniques to solve testing problems in the software is known as Search-Based Software Testing (SBST). SBST is a tool for creating test data, prioritising test cases, reducing test suites, and optimising software test oracles, among other things.	(1) It is very quick, efficient to implement and faster in execution. (2) It gives a sense of the landscape structure.	(1) Poorly defined requirements. (2) Some possible inputs will only be tested.	(Harman, 2007; Ali et al., 2010; Mairhofer et al., 2011; Devasena and Valarmathi, 2012; Varshney and Mehrotra, 2013; Rojas et al., 2015; Dave and Agrawal, 2015; Panichella, 2019; Scalabrino et al., 2021)
Specification based	Test cases are derived directly from the specification or another model of what the system should perform in a specification-based technique.	(1) Test cases are independent of implementation. (2) Development of test cases have done in parallel with the implementation.	(1) Substantial redundancies may exist in between test cases. (2) Possibility of gaps of untested software.	(Fraser and Gargantini, 2009; Brucker et al., 2011; Vasilache, 2016; Ed-Douibi et al., 2018; Wang et al., 2019; Sato, 2020)
Metamorphic Testing	Metamorphic testing technique is to test programmes without the use of an oracle has been proposed. It uses metamorphic relations, which are features of the target function, to produce follow-up test cases and automatically validate the results.	(1) Simplicity in concept. (2) Straightforward implementation.	(1) No proper framework for verification, validation, and quality assessment. (2) Metamorphic relations should not be formally described.	(Tao et al., 2010; Sun et al., 2011; Hui and Huang, 2013; Bandaru and Albert Mayan, 2016; Saha and Kanewala, 2018; Lv et al., 2018; Segura et al., 2020; Asyrofi et al., 2021)

Answer to RQ2

During the review processes it becomes clear that many of researchers have utilized various standard datasets and sample programs to validate their test case generation approaches. Table 5 shows the brief details of dataset utilized in Test Case Generation.

Table 5. Details of dataset utilized in test case generation.

Year of Publication	Dataset utilized	Title of Paper	Reference
2015	EvoSuite 10	"A Memetic Algorithm for whole test suite generation"	(Fraser et al., 2015)
2017	SF100 corpus	"A detailed investigation of the effectiveness of whole test suite generation"	(Rojas et al., 2017)
2017	Defects4J repository	"Test Case Generation for Program Repair: A Study of Feasibility and Effectiveness"	(Yu et al., 2017)
2020	Neo4j, JSON	"Test case generation based on mutations over user execution traces"	(Paiva et al., 2020)
2020	Mocha JS, Node JS	"Implementing DDD for Automatic Test Case Generation"	(Nachiengmai et al., 2020)

Answer to RQ3

The classification of research papers may help in tracking the research growth of the domain. The ratio amongst the papers published in conferences and journals of repute may give a clear idea about the growth. To provide the answer to RQ 3, authors have classified the research papers in these two classes.

Figure 7 depicts that out of 125 selected papers, 79% papers come from journal publication and 21% papers have been published in conference publication. The ratio of publication indicates that from year 2011 to 2021 maximum papers of test case generation published in a variety of journals. The lists of few papers with publication details are mentioned in the Table 8 that may help to the beginner's level researchers to explore the important literature.

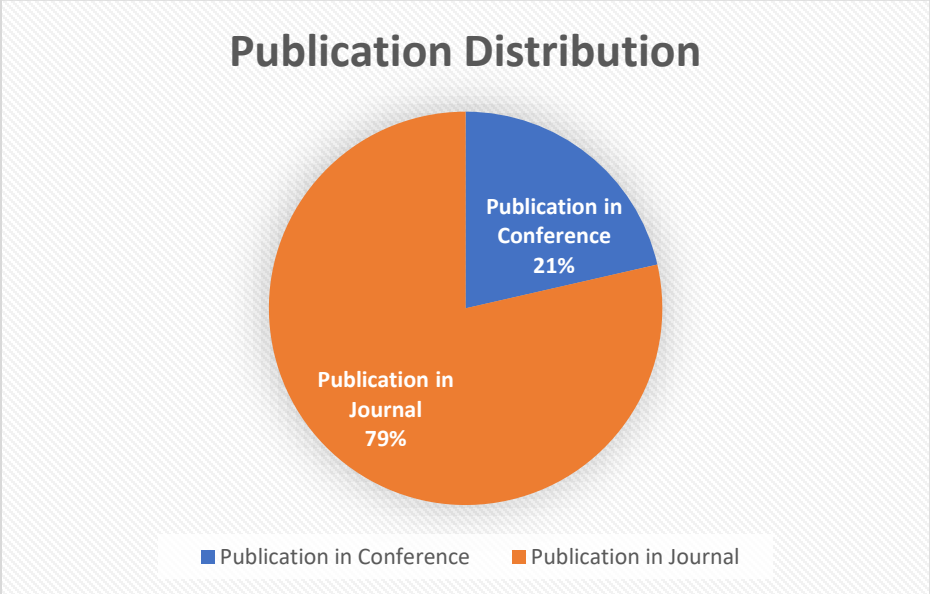

Figure 7. Publication distribution.

Answer to RQ4

The performance of any testing technique depends on test case generation. So, it is important to analyze research distribution of different types of testing solutions. To answer the RQ4, the Figure 8 has been drawn, which indicates that there are various types of testing used to conduct research in test case generation domain. It has been represented that in the field of test case generation 74% research contributes in software testing domain, 13% research contributes in the domain of regression testing, approx. 2-2% research distributes in the domain of AI testing & Evolutionary testing and 9% researchers contributes their work in other different types of testing domain like object oriented, data flow testing, mutation testing etc.

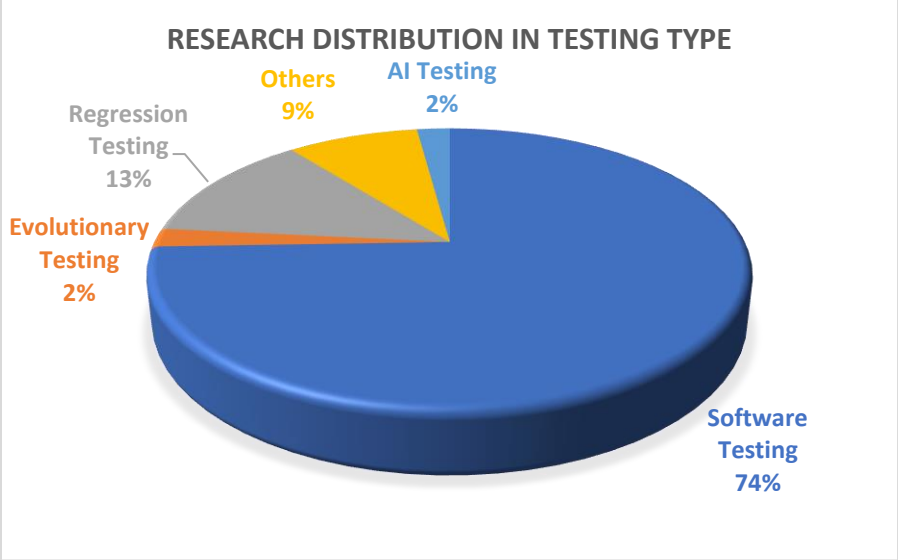

Figure 8. Research Distribution including various testing domain.

Answer to RQ5

According to literature, various testing tools have been developed to address the challenges of test case generation. In order to fulfill the requirements and maintain the quality of the software, researchers have developed a variety of tools for generating new test cases for existing/ new software. To answer the RQ5, Table 6 has been formed from the literature that list the various testing tools utilized for automatic generation of test cases. In Table 6, 24 different tools are mentioned which represent the name of tool, input language used by tool, the category of tool and short description as well as year of development/ modification of tool.

Table 6. List of various testing tools used in test case generation.

Year of Development/ Modification	Name of Tool	Input Language	Type of Software	Reference
2010	Austin	C Language	Open Source	(Lakhotia et al., 2010)
2010	PET	Java	Academic & Research	(Albert et al., 2010)
2011	Korat	Java	Open Source	(Boyapati et al., 2002)
2011	jPET	Java	Commercial	(Albert et al., 2011)
2012	Palus	Java	Open Source	(Zhang et al., 2011)
2013	MergePoint/ Mayhem	binary (32bit, Linux)	Commercial	(Avgerinos et al., 2014)

Table 6 continued…

2013	PathCrawler	C Language	Academic & Research	(Williams et al., 2005)
2014	CREST	C Language	Open Source	(Boshernitsan et al., 2006)
2014	CAUT	C Language	Academic & Research	(Su et al., 2015)
2014	Jseft	JavaScript	Academic & Open Source	(Mirshokraie et al., 2015)
2015	AgitarOne	Java	Commercial	(Burnim and Sen, 2008)
2015	AutoTest	Eiffel	Commercial & Open Source	(Leitneret al., 2007)
2015	CATG	Java	Open Source	(Tanno et al., 2015)
2015	EvoSuite	Java	Academic & Open Source	(Fraser and Arcuri, 2013)
2016	GRT	Java	Academic & Research	(Ma et al., 2016)
2015	GUITAR	GUI application	Open Source	(Nguyen et al., 2014)
2015	Jalangi	JavaScript	Open Source	(Kalasapur et al., 2013)
2015	JTExpert	Java	Academic & Research	(Sakti et al., 2015)
2015	Randoop	Java	Open Source	(Pacheco et al., 2007)
2015	Symbolic PathFinder	Java	Open Source	(Păsăreanu et al., 2013)
2015	T3	Java	Open Source	(Prasetya, 2015)
2015	TCG (LoTus)	FSM	Open Source	(Muniz et al., 2015)
2016	UML Test	XML	Open Source	(Herout and Brada, 2016)
2018	Testrecorder	Java	Open Source	(Negara et al., 2019)

Answer to RQ6

As per the selected literature, it is clear that in last decade, most of the researchers have utilized various optimization/ metaheuristic algorithms to solve test case generation problem. Table 7 gives an effective summarization of the literature and discusses various optimization algorithms applied for test case generation during the specified period of critical review. A variety of optimization algorithms including Particle Swarm Optimization, Genetic algorithms, Bee Colony Optimization, Firefly Algorithm, Ant Colony Optimization and many more have been utilized by various researchers. The Figure 8 shows that approximately 26% of the research work done in the field of test case generation by utilizing various metaheuristic/ optimization algorithms. The Figure 9 shows the utilization of various optimization in the field of test case generation.

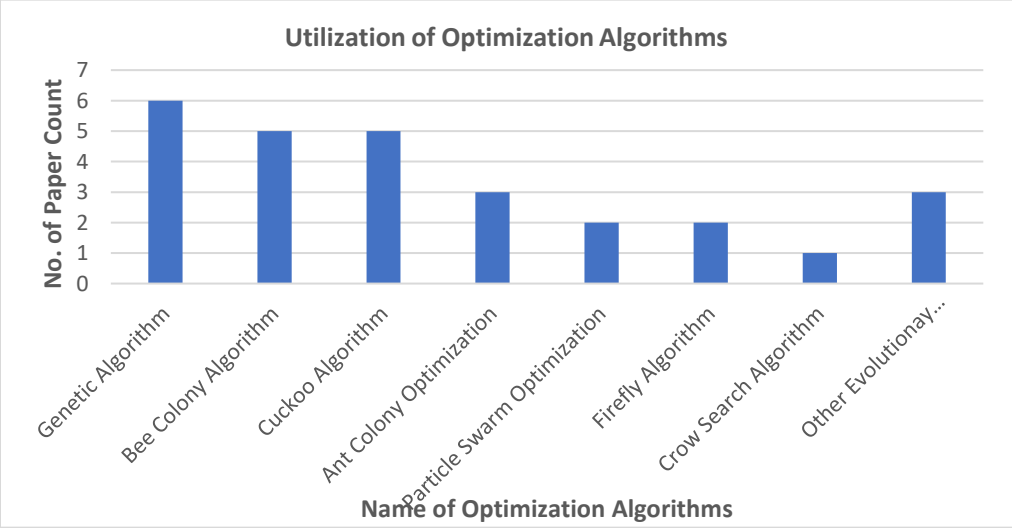

Figure 9. Utilization of various optimization algorithms.

Table 7. Various optimization algorithms used in test case generation.

Year of Publication	Name of Optimization Algorithm	Approach/ Technique	Reference
2013	Bee-Colony Optimization & Modified Genetic Algorithm	proposed an approach	(Dalal and Chhillar, 2013)
2015	Genetic Algorithms	proposed an approach using various operators	(Fraser et al., 2015)
2014	Improved environmental adaption method (IEAM)	proposed a new optimization technique	(Mishra et al., 2014)
2014	Multi-Objective Firefly Algorithm (MOFA)	proposed a technique	(Iqbal et al., 2014)
2016	Cuckoo Algorithm	For generation of test data, proposed a new application of Cuckoo search algorithm.	(Khari and Kumar, 2016)
2016	HARMONY SEARCH ALGORITHM	Harmony search optimization algorithm has been analyzed for random test data generation.	(Sahoo et al., 2016a)
2016	Genetic Algorithm	the optimization of test case generation using genetic algorithm.	(Mateen et al., 2016)
2016	Firefly Algorithm	Firefly Algorithm has been analyzed to generate random test cases & optimize the results.	(Sahoo et al., 2016)
2018	Modified Particle swarm optimization algorithm	Proposed a new technique for automatic test data generation.	(Wang and Liu, 2018)
2018	Multi-Objectives Evolutionary Algorithm approach (MOEA)	proposed an approach	(Abdallah et al., 2018)
2018	Artificial Bee Colony Algorithm	suggest a new strategy focusing on the Artificial Bee Colony (ABC) Algorithm	(Alazzawi and Rais, 2018)
2019	Cuckoo Search Algorithm	proposes a framework for the generation of an optimal set of test cases	(Sharma et al., 2019)
2020	Particle-Swarm Optimization & Adaptive Particle-Swarm Optimization	propose a fitness function, Improved Combined Fitness (ICF) function	(Sahoo and Ray, 2020)
2019	Genetic Algorithm	propose a new method for efficient test case generation	(Wang et al., 2019)
2020	Ant Colony Optimization Algorithm	proposed a technique	(Saju and Vinod 2020)
2020	Hybrid Cuckoo Search & Bee-Colony Algorithm	proposed an approach	(Lakshminarayana and SureshKumar, 2020)
2019	Ant Colony Optimization Algorithm	proposed a new self-adapting ant colony optimization-based algorithm using fuzzy logic (ACOF)	(Ahmad et al., 2020)
2020	Improved Crow Search Algorithm	proposes an Improved Crow Search Algorithm (ICSA)	(Jatana and Suri, 2020)
2020	Artificial bee colony algorithm	presented a new algorithm in the domain of data flow testing by making use of ABC optimization algorithm	(Sheoran et al., 2020)
2021	Genetic programming	proposed an approach	(Nosrati et al., 2021)
2021	Ant Colony Algorithm	developed a strategy	(Ramli et al., 2021)
2021	Artificial Bee Colony algorithm	proposed an approach Archive-based Artificial Bee Colony	(Sahin et al., 2021)
2021	Hybrid Migrating Birds Optimization, Genetic Algorithm	proposed a new strategy named as Elitist Hybrid MBO-GA Strategy (EMBO-GA)	(Zakaria et al., 2021)
2021	Adaptive cuckoo search algorithm	proposed an approach	(Sahoo et al., 2021)

4. Conclusions and Future Scope

Now a day everyone is looking for quality software product to ease their day-to-day life. Testing plays an important role to ensure the quality of software, which can be achieve though generation of testcases. Test Case Generation is a challenging task that should be more mature. In order to dig out the challenges and their solutions this paper has presented a comprehensive study of various tools and techniques for test case generation. To perform a critical review process has been followed such as search keywords, searching criteria and selection procedure. To make it more effective authors have framed various

research questions, which have been answered one by one in the result discussion section. The list of tables and figures has been presented in this study to address the answers of the research questions. It is clear from literature review that huge research work has been performed by various researches in the field of test case generation in last one decade. The authors also tried to provide a brief description of existing literature with the help of critical review and provide the answers of research questions with evidences.

During the critical review, it has been found that there are various research gaps available in existing research. Future work may include development of new testing tools as per more specific requirements or integrate the features of two or more tools to solve complex problems of test case generation process, also utilized or create more datasets for research rather than using toy or existing datasets. In future, develop new metaheuristic/ optimization algorithms to optimize the results of test case generation problems as well as increase the duration of literature in terms of years and consider maximum no. of papers for critical review in test case generation or in other domains of software testing.

Appendix

To validate the answer of Research Questions RQ2, RQ3 & RQ4, Table 8 represents the data which provide the Summary of manuscripts selected for this study and also answers of above-mentioned questions:

Table 8. Summary of manuscripts selected for this study**.**

Year	Testing	Keyword	Dataset	Approach/ Technique	Conference/ Journal	Reference
2010	Software Testing	Test Case Generation	GUI XML file	Proposed a new technique of test case generation	Conference	(Alsmadi, 2010)
2010	Software Testing	Test Case Generation	Toy Dataset	proposed a practical test case generation technique derived from use case diagram.	Conference	(Daengdej & Kosindrdecha, 2010)
2013	Software Testing	Test Case Generation	Toy Dataset	proposed a model and proposed an algorithm	Journal	(Dalal and Chhillar, 2013)
2013	Software Testing	Test Suite Generation	19 open-source libraries and programs, Toy Dataset	For test case generation, EVOSUITE tool implements the approach proposed in this paper.	Journal	(Fraser and Arcuri, 2013)
2014	Software Testing	Test Case Generation	Toy Dataset	proposed an algorithm	Journal	(Mishra et al., 2014)
2014	Software Testing	Test Case Generation	Toy Dataset	proposed a technique	Conference	(Iqbal et al., 2014)
2014	Software Testing	Test Suite Generation	No Dataset	Survey/Review	Journal	(Pahwa and Solanki, 2014)
2014	Software Testing	Test Suite Generation	No Dataset	Survey/Review	Journal	(Hooda and Chhillar, 2014)
2015	Evolutionary Testing	Test Suite Generation	EvoSuite 10 tool	Memetic Algorithm for test suite optimization	Journal	(Fraser et al., 2015)
2016	Software Testing	Test Case Generation	Toy Dataset	proposed a methodology for generating test cases	Journal	(Sahoo et al., 2016a)
2016	Software Testing	Test Case Generation	Toy Dataset	proposed an approach	Journal	(Sahoo et al., 2016)
2016	Software Testing	Test Case Generation	overview of different techniques of automatic test cases generation	Survey/ Review	Journal	(Mahadik et al., 2016)
2016	Software Testing	Test Case Generation	Toy Dataset	Proposed an optimization approach	Journal	(Mateen et al., 2016)
2016	Software Testing	Test Case Generation	Toy Dataset	paper proposed a methodology for generating test cases	Journal	(Sahoo et al., 2016b)

Table 8 continued...

2017	Software Testing	Test Case Generation	SF100 corpus	empirical study	Journal	(Rojas et al., 2017)
2017	Software Testing	Test Case Generation	Toy Dataset	Survey	Conference	(Kulshreshtha et al., 2017)
2017	Software Testing	Test Case Generation	Defects4J repository	proposed two approaches for using test case generation	Conference	(Yu et al., 2017)
2018	Software Testing	Test Case Generation	No Dataset	SLR	Journal	(Arora and Bhatia, 2018)
2018	Software Testing	Test Case Generation	Toy Dataset	proposed a new approach	Journal	(Abdallah et al., 2018)
2018	Software Testing	Test Case Generation	No Dataset	literature survey	Journal	(Chauhan et al., 2018)
2018	Regression Testing	Test Case Generation	No Dataset	Survey Paper	Journal	(Gupta et al., 2018)
2018	Software Testing	Test Cases Generation	Toy dataset	proposed a new approach	Conference	(Din and Zamli, 2018)
2019	Regression Testing	Test Suite Generation	five Java programs under test	performance evaluation of six meta- heuristic algorithms	Journal	(Khari et al., 2020)
2019	Software Testing	Test Case Generation	Toy Dataset	present a model-based approach	Journal	(Yazdani Seqerloo et al., 2019)
2019	Software Testing	Test Case Generation	No Dataset	Survey Paper	Journal	(Khari and Kumar, 2019)
2019	Software Testing	Test Case Generation	Toy Dataset	proposed an automatic approach	Journal	(Alrawashed et al., 2019)
2019	Software Testing	Test Case Generation	No Dataset	review	Journal	(Gupta et al., 2019)
2019	Software Testing	Test Case Generation	Toy dataset	proposed a method for demand-based TCG for OO systems	Journal	(Singh et al., 2019)
2019	Regression Testing	Test Case Generation	Toy dataset	propose a new method	Conference	(Wang et al., 2019)
2019	Software Testing	Test Case Generation	No Dataset	systematic review	Journal	(Mishra et al., 2019)
2019	Software Testing	Test Case Generation	10 Java programs from two open-source projects are used as case studies	Literature Review, two automatic test case generator tools are used, Randoop and Evosuite.	Conference	(Setiani et al., 2019)
2020	Software testing	Test case generation	Toy dataset	proposed a modified ACO based approach for the automated and effective generation of valid test cases	Conference	(Saju and Vinod, 2020)
2020	Software testing	Test case generation	Neo4j, JSON	presents a web testing approach in which test cases are generated from user execution traces	Journal	(Paiva et al., 2020)
2020	Software testing	Test case Generation	Toy dataset	proposed a new method CSBCA	Journal	(Lakshminarayana and SureshKumar, 2020)
2020	Software testing	Test case Generation	Toy dataset	systematic study of mapping of test case generation techniques	Journal	(Minhas et al., 2020)
2020	Regression testing	Test Case Generation	JAVA, Toy dataset	propose an innovative technique in order to test the quality of the cloud-based software	Journal	(Venkatraman and Sethumadhavan, 2020)
2020	Software testing	Automatic test case generation	Mocha JS, Node JS	Proposed a technique	Journal	(Nachiengmai et al., 2020)
2020	Software testing	Test Data Generation	Toy dataset	Proposed a Meta-heuristic technique ICSA	Journal	(Jatana and Suri, 2020)

Table 8 continued...

2020	Software testing	Test suite generation	Toy dataset	Present a novel algorithm	Journal	(Sheoran et al., 2020)
2020	Software testing	Test suite generation	Toy dataset	Proposed a DOTSG method for EFSM models using GA.	Journal	(Zhao et al., 2020)
2020	Software testing	Automated Test Suite Generation	No Dataset	Empirical Evaluation	Journal	(Khari, 2020)
2020	AI Testing	Test Case Generation	Toy dataset	two techniques used activation maximization and Generative Adversarial Network (GAN)	Journal	(Koo et al., 2020)
2021	Software testing	Test Suite Generation	Key-Points detection DNNs (KP-DNNs)	present an approach to automatically generate test data for KP-DNNs using many-objective search	Conference	(Haq et al., 2020)
2021	Software testing	Test Case Generation	No Dataset	Empirical Evaluation	Journal	(Brunetto et al., 2021)

Conflict of Interest
The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgements
This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors would like to thank the editor and anonymous reviewers for their comments that help improve the quality of this work.

References

Abdallah, S.A., Moawad, R., & Fawzy, E.E. (2018). An optimization approach for automated unit test generation tools using multi-objective evolutionary algorithms. *Future Computing and Informatics Journal*, *3*(2), 178-190. https://doi.org/10.1016/j.fcij.2018.02.004.

Alazzawi, A.K., Rais, H.M., & Basri, S. (2018). Artificial bee colony algorithm for t-way test suite generation. In *2018 4th International Conference on Computer and Information Sciences (ICCOINS)* (pp. 1-6). IEEE. Kuala Lumpur, Malaysia.

Albert, E., Cabanas, I., Flores-Montoya, A., Gómez-Zamalloa, M., & Gutiérrez, S. (2011, October). Jpet: An automatic test-case generator for java. In *2011 18th Working Conference on Reverse Engineering* (pp. 441-442). IEEE. Limerick, Ireland.

Albert, E., Gómez-Zamalloa, M., & Puebla, G. (2010). PET: a partial evaluation-based test case generation tool for Java bytecode. In *Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation* (pp. 25-28). https://doi.org/10.1145/1706356.1706363.

Ali, S., Briand, L.C., Hemmati, H., & Panesar-Walawege, R.K. (2010). A systematic review of the application and empirical investigation of search-based test case generation. *IEEE Transactions on Software Engineering*, *36*(6), 742-762. https://doi.org/10.1109/TSE.2009.52.

Alrawashed, T.A., Almomani, A., Althunibat, A., & Tamimi, A. (2019). An automated approach to generate test cases from use case description model. *CMES - Computer Modeling in Engineering and Sciences*, *119*(3), 409-425. https://doi.org/10.32604/cmes.2019.04681.

Alsmadi, I. (2010). Using genetic algorithms for test case generation and selection optimization. In *CCECE 2010* (pp. 1-4). IEEE. Calgary, AB, Canada.

Arora, P.K., & Bhatia, R. (2018). A Systematic review of agent-based test case generation for regression testing. *Arabian Journal for Science and Engineering*, *43*(2), 447-470. https://doi.org/10.1007/s13369-017-2796-4.

Asyrofi, M.H., Yang, Z., Yusuf, I.N.B., Kang, H.J., Thung, F., & Lo, D. (2021). Biasfinder: Metamorphic test generation to uncover bias for sentiment analysis systems. *IEEE Transactions on Software Engineering*, *48*(12), 5087-5107.

Avdeenko, T., & Serdyukov, K. (2021). Automated test data generation based on a genetic algorithm with maximum code coverage and population diversity. *Applied Sciences*, *11*(10), 4673. https://doi.org/10.3390/app11104673.

Avgerinos, T., Rebert, A., Cha, S.K., & Brumley, D. (2014). Enhancing symbolic execution with veritesting. *Proceedings - International Conference on Software Engineering*, *1*, 1083-1094. https://doi.org/10.1145/2568225.2568293.

Bala Mishra, D., Bilgaiyan, S., Mishra, R., Acharya, A.A., & Mishra, S. (2017). A review of random test case generation using genetic algorithm. *Indian Journal of Science and Technology*, *10*(30), 1-7. https://doi.org/10.17485/ijst/2017/v10i30/107654.

Bandaru, R., & Albert Mayan, J. (2016). Novel approach for whole test suite generation using metamorphic relations. *Indian Journal of Science and Technology*, *9*(10), 1-7. https://doi.org/10.17485/ijst/2016/v9i10/88983.

Baresi, L., & Pezze, M. (2006). An introduction to software testing. *Electronic Notes in Theoretical Computer Science*, *148*(1), 89-111. https://doi.org/10.1016/j.entcs.2005.12.014.

Bertolino, A. (2007, May). Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE'07)* (pp. 85-103). IEEE. Minneapolis, MN, USA.

Boshernitsan, M., Doong, R., & Savoia, A. (2006). From daikon to agitator: Lessons and challenges in building a commercial tool for developer testing. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis* (pp. 169-180). https://doi.org/10.1145/1146238.1146258.

Boyapati, C., Khurshid, S., & Marinov, D. (2002). Korat: Automated testing based on Java predicates. *ACM SIGSOFT Software Engineering Notes*, *27*(4), 123-133.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571-583. https://doi.org/10.1016/j.jss.2006.07.009.

Brucker, A.D., Krieger, M.P., Longuet, D., Wolff, B. (2011). A Specification-Based test case generation method for UML/OCL. In: Dingel, J., Solberg, A. (eds) *Models in Software Engineering*. *MODELS 2010. Lecture Notes in Computer Science* (vol. 6627). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21210-9_33.

Brunetto, M., Denaro, G., Mariani, L., & Pezzè, M. (2021). On introducing automatic test case generation in practice: A success story and lessons learned. *Journal of Systems and Software*, *176*, 110933. https://doi.org/10.1016/j.jss.2021.110933.

Budgen, D., & Brereton, P. (2006, May). Performing systematic literature reviews in software engineering. In *Proceedings of the 28th International Conference on Software Engineering* (pp. 1051-1052). https://doi.org/10.1145/1134285.1134500.

Burnim, J., & Sen, K. (2008, September). Heuristics for scalable dynamic test generation. In *2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (pp. 443-446). IEEE. L'Aquila, Italy.

Chauhan, A., Singhal, R.S., & Yadav, P.K. (2018). Test case generation techniques. *International Journal of Computer Trends and Technology*, *57*(2), 66-69. https://doi.org/10.14445/22312803/ijctt-v57p112.

Clark, A.G., Walkinshaw, N., & Hierons, R.M. (2021). Test case generation for agent-based models: A systematic literature review. *Information and Software Technology*, *135*, 106567. https://doi.org/10.1016/j.infsof.2021.106567.

Daengdej, J., & Kosindrdecha, N. (2010). A test case generation technique and process. *ACM SIGSOFT Software Engineering Notes*, *EMDT2010*.

Dalal, S., & Chhillar, R.S. (2013). A novel technique for generation of test cases based on bee colony optimization and modified genetic algorithm (BCOmGA). *International Journal of Computer Applications*, *68*(19), 12-16.

Dave, M., & Agrawal, R. (2015, June). Search based techniques and mutation analysis in automatic test case generation: A survey. In *2015 IEEE International Advance Computing Conference (IACC)* (pp. 795-799). IEEE. Banglore, India

de Almeida Biolchini, J.C., Mian, P.G., Natali, A.C.C., Conte, T.U., & Travassos, G.H. (2007). Scientific research ontology to support systematic review in software engineering. *Advanced Engineering Informatics*, *21*(2), 133-151. https://doi.org/10.1016/j.aei.2006.11.006.

Din, F., & Zamli, K.Z. (2018, February). Fuzzy adaptive teaching learning-based optimization strategy for GUI functional test cases generation. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications* (pp. 92-96). https://doi.org/10.1145/3185089.3185148.

Ed-Douibi, H., Izquierdo, J.L.C., & Cabot, J. (2018, October). Automatic generation of test cases for REST APIs: A specification-based approach. In *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 181-190). IEEE. Stockholm, Sweden.

Enoiu, E.P., Sundmark, D., & Pettersson, P. (2013, March). Model-based test suite generation for function block diagrams using the uppaal model checker. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops* (pp. 158-167). IEEE. Luxembourg, Luxembourg.

Fellner, A., Krenn, W., Schlick, R., Tarrach, T., & Weissenbacher, G. (2019). Model-based, mutation-driven test-case generation via heuristic-guided branching search. *ACM Transactions on Embedded Computing Systems (TECS)*, *18*(1), 1-28. https://doi.org/10.1145/3289256.

Fraser, G., & Arcuri, A. (2013). Whole test suite generation. *IEEE Transactions on Software Engineering*, *39*(2), 276-291. https://doi.org/10.1109/TSE.2012.14.

Fraser, G., Arcuri, A., & McMinn, P. (2015). A memetic algorithm for whole test suite generation. *Journal of Systems and Software*, *103*, 311-327. https://doi.org/10.1016/j.jss.2014.05.032.

Fraser, G., & Gargantini, A. (2009, May). Experiments on the test case length in specification based test case generation. In *2009 ICSE Workshop on Automation of Software Test* (pp. 18-26). IEEE. Vancouver, BC, Canada.

Gupta, N., Sharma, A., & Pachariya, M.K. (2019). An insight into test case optimization: ideas and trends with future perspectives. *IEEE Access*, *7*, 22310-22327. https://doi.org/10.1109/ACCESS.2019.2899471.

Gupta, N., Yadav, V., & Singh, M. (2018). Automated regression test case generation for web application: A survey. *ACM Computing Surveys (CSUR)*, *51*(4), 1-25. https://doi.org/10.1145/3232520.

Haq, F.U., Shin, D., Briand, L.C., Stifter, T., & Wang, J. (2020). Automatic test suite generation for key-points detection dnns using many-objective search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '21), Virtual, Denmark* (Vol. 1, Issue 1). Association for Computing Machinery. https://doi.org/10.1145/3460319.3464802.

Harman, M. (2007, May). Automated test data generation using search based software engineering. In *Second International Workshop on Automation of Software Test (AST'07)* (pp. 2-2). IEEE. Minneapolis, MN, USA.

Herout, P., & Brada, P. (2016, April). Uml-test application for automated validation of students' UML class diagram. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)* (pp. 222-226). IEEE. Dallas, TX, USA.

Hooda, I., & Chhillar, R. (2014). A review: Study of test case generation techniques. *International Journal of Computer Applications*, *107*(16), 33-37. https://doi.org/10.5120/18839-0375.

Huang, R., Xie, X., Chen, T.Y., & Lu, Y. (2012). Adaptive random test case generation for combinatorial testing. In *2012 IEEE 36th Annual Computer Software and Applications Conference* (pp. 52-61). IEEE. Izmir, Turkey.

Hui, Z.W., & Huang, S. (2013, December). Achievements and challenges of metamorphic testing. In *2013 Fourth World Congress on Software Engineering* (pp. 73-77). IEEE. Hong Kong, China.

Iqbal, N., Zafar, K., & Zyad, W. (2014, April). Multi-objective optimization of test sequence generation using multi-objective firefly algorithm (MOFA). In *2014 International Conference on Robotics and Emerging Allied Technologies in Engineering (iCREATE)* (pp. 214-220). IEEE. Islamabad, Pakistan.

Jatana, N., & Suri, B. (2020). An improved crow search algorithm for test data generation using search-based mutation testing. *Neural Processing Letters*, *52*(1), 767-784.

Khari, M. (2020). Empirical evaluation of automated test suite generation and optimization. *Arabian Journal for Science and Engineering*, *45*(4), 2407-2423. https://doi.org/10.1007/s13369-019-03996-3.

Khari, M., & Kumar, P. (2016, March). A novel approach for software test data generation using cuckoo algorithm. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies* (pp. 1-6). https://doi.org/10.1145/2905055.2905157.

Khari, M., & Kumar, P. (2019). An extensive evaluation of search-based software testing: a review. *Soft Computing*, *23*(6), 1933-1946. https://doi.org/10.1007/s00500-017-2906-y.

Khari, M., Sinha, A., Verdú, E., & Crespo, R.G. (2020). Performance analysis of six meta-heuristic algorithms over automated test suite generation for path coverage-based optimization. *Soft Computing*, *24*(12), 9143-9160. https://doi.org/10.1007/s00500-019-04444-y.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and Software Technology*, *51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009.

Koo, B., Bae, J., Kim, S., Park, K., & Kim, H. (2020). Test case generation method for increasing software reliability in safety-critical embedded systems. *Electronics (Switzerland)*, *9*(5), 1-16. https://doi.org/10.3390/electronics9050797.

Kulshreshtha, M., Agarwal, C., & Kamalakannan, J. (2017, November). Comparative study on test case generation: a survey. In *IOP Conference Series: Materials Science and Engineering* (Vol. 263, No. 4, p. 042035). IOP Publishing. https://doi.org/10.1088/1757-899X/263/4/042035.

Lakhotia, K., Harman, M., & Gross, H. (2010, September). AUSTIN: A tool for search based software testing for the C language and its evaluation on deployed automotive systems. In *2nd International Symposium on Search based Software Engineering* (pp. 101-110). IEEE. Benevento, Italy.

Lakshminarayana, P., & SureshKumar, T.V. (2020). Automatic generation and optimization of test case using hybrid cuckoo search and bee colony algorithm. *Journal of Intelligent Systems*, *30*(1), 59-72. https://doi.org/10.1515/jisys-2019-0051.

Lemberger, T. (2020). Plain random test generation with PRTest. *International Journal on Software Tools for Technology Transfer*, 1-3. https://doi.org/10.1007/s10009-020-00568-x.

Li, W., Le Gall, F., & Spaseski, N. (2018). A survey on model-based testing tools for test case generation. In *International Conference on Tools and Methods for Program Analysis* (pp. 77-89). Springer, Cham.

Liu, Z., Gao, X., & Long, X. (2010). Adaptive random testing of mobile application. In *2010 2nd International Conference on Computer Engineering and Technology* (Vol. 2, pp. V2-297). IEEE. Chengdu, China.

Lv, X.W., Huang, S., Hui, Z.W., & Ji, H.J. (2018). Test cases generation for multiple paths based on PSO algorithm with metamorphic relations. *IET Software*, *12*(4), 306-317. https://doi.org/10.1049/iet-sen.2017.0260.

Ma, L., Artho, C., Zhang, C., Sato, H., Gmeiner, J., & Ramler, R. (2015, November). Grt: Program-analysis-guided random testing (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 212-223). IEEE. Lincoln, NE, USA.

Mahadik, P., Bhattacharyya, D., & Kim, H.J. (2016). Techniques for automated test cases generation: A review. *International Journal of Software Engineering and Its Applications*, *10*(12), 13-20. https://doi.org/10.14257/ijseia.2016.10.12.02

Mairhofer, S., Feldt, R., & Torkar, R. (2011, July). Search-based software testing and test data generation for a dynamic programming language. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (pp. 1859-1866). https://doi.org/10.1145/2001576.2001826.

Mateen, A., Nazir, M., & Afsar, S. (2016). Optimization of test case generation using genetic algorithm (GA). *International Journal of Computer Applications*, *151*(7), 6-14. https://doi.org/10.5120/ijca2016911703.

Leitner, A., Ciupa, I., Meyer, B., & Howard, M. (2007, January). Reconciling manual and automated testing: The autotest experience. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)* (pp. 261a-261a). IEEE. Waikoloa, HI, USA.

Minhas, N.M., Masood, S., Petersen, K., & Nadeem, A. (2020). A systematic mapping of test case generation techniques using UML interaction diagrams. *Journal of Software: Evolution and Process*, *32*(6), 1-21. https://doi.org/10.1002/smr.2235.

Mirshokraie, S., Mesbah, A., & Pattabiraman, K. (2015, April). JSeft: Automated JavaScript unit test generation. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)* (pp. 1-10). IEEE. Graz, Austria.

Mishra, D.B., Acharya, A.A., & Mishra, R. (2019). Evolutionary algorithms for path coverage test data generation and optimization: A review. *Indonesian Journal of Electrical Engineering and Computer Science*, *15*(1), 504-510. https://doi.org/10.11591/ijeecs.v15.i1.pp504-510.

Mishra, K.K., Tiwari, S., & Misra, A.K. (2014). Improved environmental adaption method and its application in test case generation. *Journal of Intelligent and Fuzzy Systems*, *27*(5), 2305-2317. https://doi.org/10.3233/IFS-141195.

Muniz, L., Netto, U.S., & Maia, P.H.M. (2015). A model-based testing tool for functional and statistical testing. In *Proceedings of the 17th International Conference on Enterprise Information Systems (ICEIS)* (pp. 404-411). https://doi.org/10.5220/0005398604040411.

Nabuco, M., & Paiva, A.C. (2014, June). Model-based test case generation for web applications. In *Int*ernational *Conference on Computational Science and its Application*s (pp. 248-262). Springer, Cham. https://doi.org/10.1007/978-3-319-09153-2_19.

Nachiengmai, W., Ramingwong, S., & Kongkeaw, A. (2020). Implementing DDD for automatic test case generation. *International Journal of Information and Education Technology*, *10*(2), 117-121. https://doi.org/10.18178/ijiet.2020.10.2.1349

Narciso, E.N., Delamaro, M.E., & De Lourdes, F.D.S.N. (2014). Test case selection: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering*, *24*(4), 653-676. https://doi.org/10.1142/S0218194014500259.

Negara, S., Esfahani, N., & Buse, R. (2019, May). Practical android test recording with espresso test recorder. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 193-202). IEEE. Montreal, QC, Canada.

Nguyen, B.N., Robbins, B., Banerjee, I., & Memon, A. (2014). GUITAR: An innovative tool for automated testing of GUI-driven software. *Automated Software Engineering*, *21*(1), 65-105. https://doi.org/10.1007/s10515-013-0128-9.

Nguyen, C.D., Marchetto, A., & Tonella, P. (2012, July). Combining model-based and combinatorial testing for effective test case generation. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis* (pp. 100-110). https://doi.org/10.1145/04000800.2336765.

Nosrati, M., Haghighi, H., & Asl, M.V. (2021). Test data generation using genetic programming. *Information and Software Technology*, *130*, 106446. https://doi.org/10.1016/j.infsof.2020.106446.

Pacheco, C., Lahiri, S.K., Ernst, M.D., & Ball, T. (2007, May). Feedback-directed random test generation. In *29th International Conference on Software Engineering (ICSE'07)* (pp. 75-84). IEEE. Minneapolis, MN, USA.

Pahwa, N., & Solanki, K. (2014). UML based Test case generation methods: A review. *International Journal of Computer Applications*, *95*(20), 1-6. https://doi.org/10.5120/16707-6859.

Paiva, A.C., Restivo, A., & Almeida, S. (2020). Test case generation based on mutations over user execution traces. *Software Quality Journal*, *28*(3), 1173-1186. https://doi.org/10.1007/s11219-020-09503-4.

Panichella, A. (2019, May). Beyond unit-testing in search-based test case generation: Challenges and opportunities. In *2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)* (pp. 7-8). IEEE. Montreal, QC, Canada.

Pasareanu, C.S., Schumann, J., Mehlitz, P., Lowry, M., Karsai, G., Nine, H., & Neema, S. (2009, July). Model based analysis and test generation for flight software. In *2009 Third IEEE International Conference on Space Mission Challenges for Information Technology* (pp. 83-90). IEEE. Pasadena, CA, USA.

Păsăreanu, C. S., Visser, W., Bushnell, D., Geldenhuys, J., Mehlitz, P., & Rungta, N. (2013). Symbolic PathFinder: Integrating symbolic execution with model checking for Java bytecode analysis. *Automated Software Engineering*, *20*(3), 391-425. https://doi.org/10.1007/s10515-013-0122-2

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2007). Systematic mapping studies in software engineering. *International Journal of Software Engineering & Knowledge Engineering*, *17*(1), 33-55.

Prasanna, M., Chandran, K.R., & Thiruvenkadam, K. (2011). Automatic test case generation for UML collaboration diagrams. *IETE Journal of Research*, *57*(1), 77-81. https://doi.org/10.4103/0377-2063.78373.

Prasetya, I.W.B. (2015, August). T3i: A tool for generating and querying test suites for java. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (pp. 950-953). https://doi.org/10.1145/2786805.2803182.

Ramler, R., Winkler, D., & Schmidt, M. (2012, September). Random test case generation and manual unit testing: Substitute or complement in retrofitting tests for legacy code?. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications* (pp. 286-293). IEEE. Cesme, Turkey.

Ramli, N., Othman, R.R., Hendradi, R., & Iszaidy, I. (2021, February). T-way test suite generation strategy based on ant colony algorithm to support t-way variable strength. In *Journal of Physics: Conference Series* (Vol. 1755, No. 1, p. 012034). IOP Publishing. https://doi.org/10.1088/1742-6596/1755/1/012034.

Rojas, J.M., Campos, J., Vivanti, M., Fraser, G., & Arcuri, A. (2015, September). Combining multiple coverage criteria in search-based unit test generation. In *International Symposium on Search Based Software Engineering* (pp. 93-108). Springer, Cham. https://doi.org/10.1007/978-3-319-22183-0_7.

Rojas, J.M., Vivanti, M., Arcuri, A., & Fraser, G. (2017). A detailed investigation of the effectiveness of whole test suite generation. *Empirical Software Engineering*, *22*(2), 852-893. https://doi.org/10.1007/s10664-015-9424-2.

Devasena, M.G., & Valarmathi, M.L. (2012). Search based Software testing technique for structural test case generation. *International Journal of Applied Information Systems*, *1*(6), 20-25. https://doi.org/10.5120/ijais12-450185.

Saju, S.S., Vinod, C.S.S. (2020). An ant colony optimization algorithm based automated generation of software test cases. In: Tan, Y., Shi, Y., Tuba, M. (eds) *Advances in Swarm Intelligence.* ICSI 2020. Lecture Notes in Computer Science (vol 12145). Springer, Cham. https://doi.org/10.1007/978-3-030-53956-6_21.

Saha, P., & Kanewala, U. (2018, May). Fault detection effectiveness of source test case generation strategies for metamorphic testing. In *Proceedings of the 3rd International Workshop on Metamorphic Testing* (pp. 2-9). https://doi.org/10.1145/3193977.3193982.

Sahin, O., Akay, B., & Karaboga, D. (2021). Archive-based multi-criteria artificial bee colony algorithm for whole test suite generation. *Engineering Science and Technology, an International Journal*, *24*(3), 806-817. https://doi.org/10.1016/j.jestch.2020.12.011.

Sahoo, R.K., Mohapatra, D.P., & Patra, M.R. (2016). A firefly algorithm based approach for automated generation and optimization of test cases. *International Journal of Computer Sciences and Engineering*, *4*(8), 1-6.

Sahoo, R.K., Ojha, D., Mohapatra, D.P., & Patra, M.R. (2016a). Automatic generation and optimization of test data using harmony search algorithm. *Computer Science & Information Technology*, *23*(10.5121).

Sahoo, R.K., Ojha, D., Mohapatra, D.P., & Patra, M.R. (2016b). Automated test case generation and optimization: a comparative review. *International Journal of Computer Science & Information Technology*, *8*(5), 19-32. https://doi.org/10.5121/ijcsit.2016.8502.

Sahoo, R.K., Satpathy, S., Sahoo, S., & Sarkar, A. (2021). Model driven test case generation and optimization using adaptive cuckoo search algorithm. *Innovations in Systems and Software Engineering*, *18*(2), 321-331. https://doi.org/10.1007/s11334-020-00378-z.

Sahoo, R.R., & Ray, M. (2020). PSO based test case generation for critical path using improved combined fitness function. *Journal of King Saud University-Computer and Information Sciences*, *32*(4), 479-490. https://doi.org/10.1016/j.jksuci.2019.09.010.

Sakti, A., Pesant, G., & Guéhéneuc, Y.G. (2015). Instance generator and problem representation to improve object oriented code coverage. *IEEE Transactions on Software Engineering*, *41*(3), 294-313. https://doi.org/10.1109/TSE.2014.2363479.

Sato, Y. (2020, December). Specification-based test case generation with constrained genetic programming. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 98-103). IEEE. Macau, China.

Scalabrino, S., Mastropaolo, A., Bavota, G., & Oliveto, R. (2021). An adaptive search budget allocation approach for search-based test case generation. *ACM Transactions on Software Engineering and Methodology*, *30*(3), 1-26. https://doi.org/10.1145/3446199.

Segura, S., Towey, D., Zhou, Z.Q., & Chen, T.Y. (2020). Metamorphic testing: Testing the untestable. *IEEE Software*, *37*(3), 46-53. https://doi.org/10.1109/MS.2018.2875968.

Sen, K., Kalasapur, S., Brutch, T., & Gibbs, S. (2013, August). Jalangi: A selective record-replay and dynamic analysis framework for JavaScript. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering* (pp. 488-498). https://doi.org/10.1145/2491411.2491447.

Setiani, N., Ferdiana, R., Santosa, P.I., & Hartanto, R. (2019, January). Literature review on test case generation approach. In *Proceedings of the 2nd International Conference on Software Engineering and Information Management* (pp. 91-95). https://doi.org/10.1145/3305160.3305186.

Sharma, S., Rizvi, S.A.M., & Sharma, V. (2019, January). A framework for optimization of software test cases generation using cuckoo search algorithm. In *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 282-286). IEEE. Noida, India.

Sheoran, S., Mittal, N., & Gelbukh, A. (2020). Artificial bee colony algorithm in data flow testing for optimal test suite generation. *International Journal of Systems Assurance Engineering and Management*, *11*(2), 340-349. https://doi.org/10.1007/s13198-019-00862-1.

Shirole, M., & Kumar, R. (2013). UML behavioral model based test case generation. *ACM SIGSOFT Software Engineering Notes*, *38*(4), 1-13. https://doi.org/10.1145/2492248.2492274.

Singh, R., Bhatia, R., & Singhrova, A. (2019). Demand based test case generation for object oriented system. *IET Software*, *13*(5), 403-413. https://doi.org/10.1049/iet-sen.2018.5043.

Su, T., Fu, Z., Pu, G., He, J., & Su, Z. (2015, May). Combining symbolic execution and model checking for data flow testing. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 654-665). IEEE. Florence, Italy.

Sun, C.A., Wang, G., Mu, B., Liu, H., Wang, Z., & Chen, T.Y. (2011, July). Metamorphic testing for web services: Framework and a case study. In *2011 IEEE International Conference on Web Services* (pp. 283-290). IEEE. Washington, DC, USA.

Sun, C.A., Wang, G., Mu, B., Liu, H., Wang, Z., & Chen, T.Y. (2011, July). Metamorphic testing for web services: Framework and a case study. In *2011 IEEE International Conference on Web Services* (pp. 283-290). IEEE. Washington, DC, USA.

Tanno, H., Zhang, X., Hoshino, T., & Sen, K. (2015, May). TESMA and CATG: automated test generation tools for models of enterprise applications. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 2, pp. 717-720). IEEE. Florence, Italy.

Tao, Q., Wu, W., Zhao, C., & Shen, W. (2010, November). An automatic testing approach for compiler based on metamorphic testing technique. In *2010 Asia Pacific Software Engineering Conference* (pp. 270-279). IEEE. Sydney, NSW, Australia.

Varshney, S., & Mehrotra, M. (2013). Search based software test data generation for structural testing. *ACM SIGSOFT Software Engineering Notes*, *38*(4), 1-6. https://doi.org/10.1145/2492248.2492277.

Vasilache, S. (2016). Specification-based test case generation using dependency diagrams. In *Proceedings of the World Congress on Engineering and Computer Science 2016* (pp. 185-189). San Francisco, USA.

Venkatraman, P., & Sethumadhavan, G. (2020). Regression testing in green cloud based software with the aid of hybrid PSO-CS algorithm. *Journal of Green Engineering*, *10*(2), 360-375.

Wang, R., Sato, Y., & Liu, S. (2019, June). Specification-based Test Case Generation with Genetic Algorithm. In *2019 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1382-1389). IEEE. Wellington, New Zealand.

Wang, Z., & Liu, Q. (2018, August). A software test case automatic generation technology based on the modified particle swarm optimization algorithm. In *2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)* (pp. 156-159). IEEE. Hunan, China.

Wetzlmaier, T., & Ramler, R. (2017, September). Hybrid monkey testing: Enhancing automated GUI tests with random test generation. In *Proceedings of the 8th ACM SIGSOFT International Workshop on Automated Software Testing* (pp. 5-10). https://doi.org/10.1145/3121245.3121247.

Williams, N., Marre, B., Mouy, P., & Roger, M. (2005, April). Pathcrawler: Automatic generation of path tests by combining static and dynamic analysis. In *European Dependable Computing Conference* (pp. 281-292). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11408901_21.

Yazdani Seqerloo, A., Amiri, M.J., Parsa, S., & Koupaee, M. (2019). Automatic test cases generation from business process models. *Requirements Engineering*, *24*(1), 119-132. https://doi.org/10.1007/s00766-018-0304-3.

Yu, Z., Martinez, M., Danglot, B., Durieux, T., & Monperrus, M. (2017). Test case generation for program repair: A study of feasibility and effectiveness. *arXiv preprint arXiv:1703.00198*.

Ahmad, M.Z.Z., Othman, R.R., Ali, M.S.A.R., & Ramli, N. (2020, February). A self-adapting ant colony optimization algorithm using fuzzy logic (ACOF) for combinatorial test suite generation. In *IOP Conference Series: Materials Science and Engineering* (Vol. 767, No. 1, p. 012017). IOP Publishing. https://doi.org/10.1088/1757-899X/767/1/012017.

Zakaria, H.L., Zamli, K.Z., & Din, F. (2021, April). Hybrid migrating birds optimization strategy for t-way test suite generation. In *Journal of Physics: Conference Series* (Vol. 1830, No. 1, p. 012013). IOP Publishing. https://doi.org/10.1088/1742-6596/1830/1/012013.

Zhang, S., Saff, D., Bu, Y., & Ernst, M.D. (2011, July). Combined static and dynamic automated test generation. In *Proceedings of the 2011 International Symposium on Software Testing and Analysis* (pp. 353-363). https://doi.org/10.1145/2001420.2001463.

Zhao, R., Wang, W., Song, Y., & Li, Z. (2020). Diversity-oriented test suite generation for EFSM model. *IEEE Transactions on Reliability*, *69*(2), 611-631. https://doi.org/10.1109/TR.2020.2971095.

Zhou, Z. Q. (2010, July). Using coverage information to guide test case selection in adaptive random testing. In *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops* (pp. 208-213). IEEE. Seoul, Korea.

Zhu, H., Hall, P.A.V., & May, J.H.R. (1997). Software unit test coverage and adequacy. *ACM Computing Surveys*, *29*(4), 366-427. https://doi.org/10.1145/267580.267590.