Metaheuristic Techniques for Test Case Optimization: A Systematic Literature Review

James Maina Mburu *jmburu48@gmail.com*
Department of Information Technology
Murang'a University of Technology
Murang'a, 75-10200, Kenya

John Gichuki Ndia *jndia@mut.ac.ke*
Department of Information Technology
Murang'a University of Technology
Murang'a, 75-10200, Kenya

Samson Wanjala Munialo *smunialo@must.ac.ke*
Department of Information Technology
Meru University of Science and Technology
Meru, 972-60200, Kenya

Abstract

Test case production is a crucial phase in the software testing lifecycle that consumes significant time, effort, and cost. As such, it is considered an optimization problem that can be addressed using metaheuristic techniques. This study aims to identify the metaheuristic techniques and their parameters used to generate optimal test data, the Unified Modeling Language (UML) diagrams and intermediate formats employed to create test cases, as well as the databases and metrics used to evaluate the performance of these techniques. A total of 46 primary studies published between 2010 and 2023 were reviewed, selected from an initial pool of 424 articles sourced from IEEE, Springer, Elsevier, and Google Scholar. The findings indicate that both single and hybrid metaheuristic techniques have been applied for test case optimization; however, the majority of studies employed single techniques, with Genetic Algorithms being the most frequently used. Furthermore, 50% of the studies did not specify the parameters used, while those that did often lacked proper documentation and failed to address the crucial balance between exploration and exploitation factors. Moreover, most studies (35) applied individual UML diagrams, mainly activity diagrams, while only 11 studies utilized multiple UML diagrams. Additionally, Graphs were the predominant intermediate format, used in 83% of the studies, whereas formats like XML, adjacency matrices, and tree structures were rarely considered. In terms of performance evaluation, most studies (21) utilized the ATM database, while 18 studies employed simple programs. Finally, while the majority of studies focused on metrics for evaluating the effectiveness of the techniques, only a few considered metrics related to efficiency (RQ6). To address these gaps, future research should consider expert opinion surveys to identify key parameters that ensure an optimal balance between exploration and exploitation. Also, future techniques should support the generation of test cases from multiple UML diagrams. The performance of these techniques should be evaluated through comparative studies using large databases, with equal emphasis on both effectiveness and efficiency metrics.

Keywords: Software Testing, Test Case Production, Metaheuristic Techniques, Optimization, UML.

1. INTRODUCTION

Test Case Production (TCP) is a significant step of software testing. Automating this process can minimize time, effort and cost of software testing. However it is a challenging and complex task

(Rao, 2016). Metaheuristic techniques play a critical function in automatic or semiautomatic creation of suitable test suite for software. The main goal of evolutionary testing is to attain high degree of automation with quality tests at a low cost. TCP guarantees that, software delivered is bug free and is of high value. Conversely, optimization ensures that ideal test case is created (Lakshminarayana & Sureshkumar, 2020; Mburu & Ndia, 2022).

There are different techniques applied to produce test case that include such as model based approaches which produce the test case from the UML models (Mburu et al., 2020; Fan et al., 2021), search-based test production which uses metaheuristic techniques that direct the exploration towards the possible areas of input space (Cuong-le et al., 2021, Alzaqebah et al., 2021), random approaches that creates test paths based on conventions, Goal based test case creation approach that covers a specific segment, statement or function (Hasan et al., 2025), and specification based techniques that create test case based on the formal requirement specifications (Aditi et al., 2025).

Over recent years, various metaheuristic techniques have been introduced such as Cuckoo Search (CS) Technique (Li et al., 2020; Sahoo, Satpathy, et al., 2021; Xiong et al., 2023), hybrid of CS and Bee Colony (BC) techniques (Lakshminarayana & Sureshkumar, 2020), a hybrid of genetic-based crow search algorithms (GBCA) (Tamizharasi & Ezhumalai, 2022), a hybrid of Firefly (FA) and BC techniques (Panigrahi et al., 2021), Genetic Algorithm (GA) (Sahoo, Derbali, Jerbi, & Thang, 2021). These techniques have been employed in creating and optimizing test case. However, they are faced by myriad challenges as they are characterized by various numbers of iterations thus spend a lot of time in selecting the required test cases. In addition, they produce and optimize from either one or two UML diagrams (Sahoo, Satpathy, et al., 2021; Panigrahi et al., 2021; Tatale & Prakash, 2022) hence they are not efficient in terms of test coverage. The said techniques are therefore, challenged for their application in TCP and optimization.

Model based testing is an approach which is used for designing and modeling the artifacts of the software. In this study, a model depicts the function (behavior) of software under test and a function can be in terms of input, output, action, events and many more. Software testing relies on the models as the test case remains the same even after certain changes are made in the code. Test cases are created from the model that defines the behavior of the software (Panda et al., 2020; Abayatilake & Blessing, 2021; Mohd-Shafie et al., 2022).

UML is a modeling language employed to visualize, examine and document the parts of a system in form of a model or design. The UML models are categorized into two; the structural models and behavioral models. Structural models define the structure of the software and represent the static aspect of the system, while the behavioral models describe the dynamic features of the software (Panigrahi et al., 2021; Wambui et al., 2024).

The study intended to presents an overview of state-of-the-art research on different metaheuristic techniques for TCP and optimization, their parameters, UML models, intermediate formats, and evaluation of the techniques and the research gaps of the study. The study was done using systematic literature review (SLR) protocol presented in Section 2 and it covered hundreds of scientific publications from goggle scholar, IEEE digital library and springer.

The rest of the paper is organized as follows; the protocol for the SLR applied to identify and assess papers in this study is described in section 2, the results of the study are presented in Section 3, in section 4, possible threats to the validity of this study are deliberated, and lastly in Section 5, we present our conclusion.

2. RESEARCH METHOD

This study employed a Systematic Literature Review (SLR) methodology, a formal method in Evidence-Based Software Engineering (EBSE), as proposed by Kitchenham et al. (2009, 2010).

The SLR methodology is grounded in a deductive research approach, where the investigation begins with predefined objectives and research questions based on existing theories and knowledge gaps, and proceeds to systematically collect and analyze evidence from the literature. To begin with, a review protocol was established (see Fig. 1), outlining the study's objectives, formulated research questions, defined search strategy, inclusion and exclusion criteria for study selection, and the construction of the search strings, as detailed below.

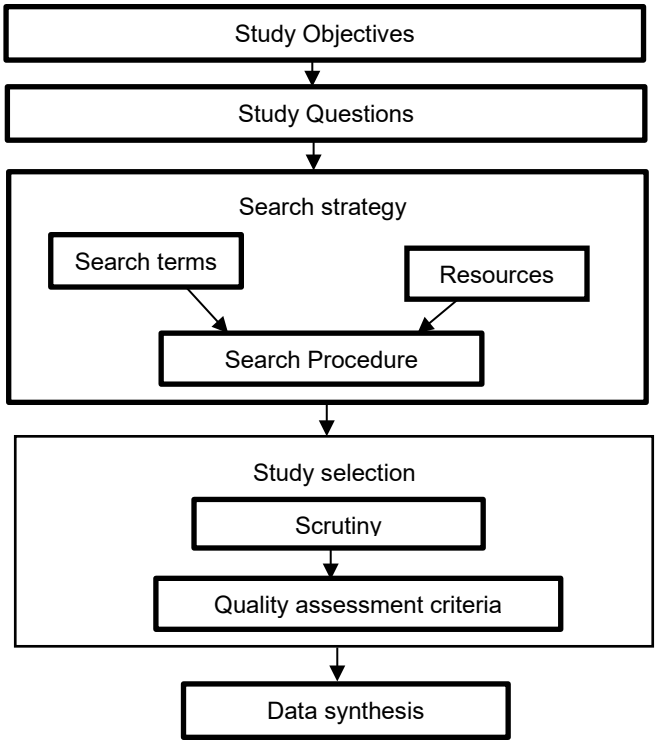

FIGURE 1: The stages of review study.

2.1 Review Protocol

An organized review protocol expresses the reason, assumption, and arranging approaches for the review. It is planned before a review is started and applied as a guide to perform the review. Fig 1 indicates the stages of review, such as objectives, research questions, search strategy, study selection, and data analysis.

2.2 Study Objectives

This study entails the following objectives;

- To determine the existing metaheuristic techniques and their parameters employed for test case optimization.
- To identify the UML behavioral models and intermediate formats used by metaheuristic techniques for TCP.
- To determine databases and metrics employed to evaluate the performance metaheuristic techniques
- To determine gaps in the present studies.
- Suggest future work about enhancement on TCP and optimization.

2.3 Study Questions

Research questions are critical part of system review as suggested by Kitchenham et al.(2010). To achieve the objectives mentioned in (section 2.2), researchers identified the research question;

RQ1: How are metaheuristic techniques employed for test case optimization?
RQ2: Which are the common parameters used by metaheuristic techniques for test case optimization?
RQ3: How are UML models applied by metaheuristic techniques for TCP?
RQ4: Which intermediate format is used by metaheuristic techniques for TCP?
RQ5: Which databases are employed to evaluate the performance of metaheuristic techniques?
RQ6: Which metrics are used to evaluate the performance meta heuristic techniques?

2.4 Search Terms

Table 1 shows the search terms used when searching for original papers for this study. The search terms are derived from the research questions.

Term	Alternate spelling
Model-based	Model-driven
Metaheuristic	
Techniques	Algorithms
Test case	Test cases
Generation	-
Optimization	-
UML	Unified modeling language
Diagram	Diagrams

TABLE 1: The search terms for searching original studies.

2.5 Search Strings

The search terms listed in Table 1 were combined into two search strings for use in the digital libraries. These are shown in Table 2.

No	Search String
1	Model-Based AND Metaheuristic AND Techniques AND (Test case OR Test cases) AND Generation AND Optimization AND (UML OR Unified Modeling Language) AND (Diagram OR Diagrams)
2	Model-Driven AND Metaheuristic AND Algorithms AND (Test case OR Test cases) AND Generation AND Optimization AND (UML OR Unified Modeling Language) AND (Diagram OR Diagrams)

TABLE 2: The search strings for the digital libraries.

2.6 Searching Strategy

During phase 1 of the search, the research study searched from IEEE, Springer, Elsevier and Google scholar. The research study then perused through the abstract of studies after which identified articles were downloaded from the electronic databases indicated in Figure. 2.

A total of 424 studies as indicates in Fig. 2 found linked to the study area during the publishing years 2010–2023. These studies are published in different digital Libraries such as Google scholar, IEEE, Springer and Elsevier. Figure 2 indicates the number of published papers in the journal and conference on model-based TCP and optimization.

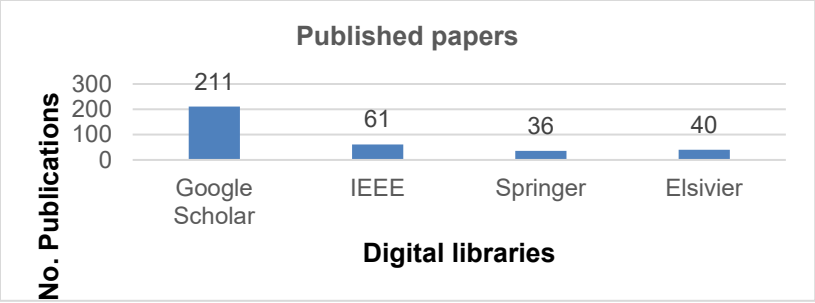

FIGURE 2: Number of publications per database.

2.7 Studies Selection Procedure

The suitable studies were selected based on the following study inclusion and exclusion criteria as represented in Table 3. After applying these measures, some papers were detached from the pool as these papers were not fulfilling the search conditions of review procedure.

Type selection conditions	ID	Description
Inclusion conditions	ICI	Studies that include topics of model-based bio-inspired metaheuristic techniques for test case generation and optimization
	ICII	Articles published between 2010-2023
	ICIII	Articlesavailable in either journals or conference proceedings
Exclusion conditions	ECI	Papers not written in English language
	ECII	Papers relating to structural(code) testing
	ECIII	Paper relating to surveys/ SLRs
	ECIV	Books, reports, thesis and tutorials
	ECV	Duplicate papers from different resources

TABLE 3: Inclusion and exclusion criteria for selecting the appropriate studies.

Figure 3 indicates the number of search phrases and considered studies at each phase. In phase 1, the search was done on digital libraries mentioned in section 2.6 by employing search terms specified in section 2.5. The search was on the base of titles, abstracts and keywords of the research studies. A total of 424 studies were obtained by the researchers. Numerous of the studies were extraneous and not quite addressed the research questions of the review. Therefore, in stage 2, researchers removed duplicate studies and studies that were not written in English language. As a result, 393 studies were obtained. At phase 3, the studies were categorized into journal papers, conference papers, thesis, technical reports and book chapters. Thesis book chapters and technical reports were discarded and a total of 354 studies were obtained. In phase 4, researchers applied the quality assessment criteria specified in section 3.8 and finally, a total of 46 studies were selected.

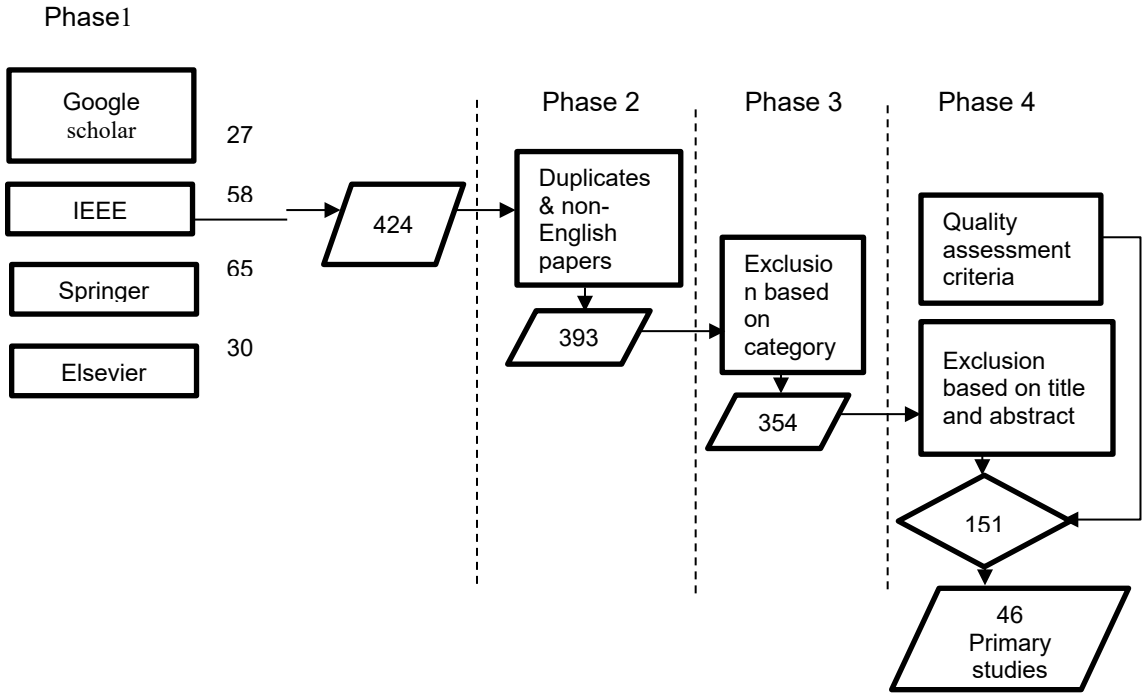

FIGURE 3: Phases of search strategy.

2.8 Study Quality Assessment Checklist and Procedure

The selected papers were evaluated based on their quality in terms of contribution to TCP and optimization. Two researchers evaluated the quality of the selected papers with one researcherevaluating all independently, while the other researcher assessed the half of the papers. Table 4 shows a checklist for evaluating value of the study.

NO	Questions
	Theoretical Contribution
1	Is at least one of the questions addressed?
2	Was the study designed to address some of the research questions?
3	Is a problem description for the research explicitly provided?
4	Is the problem description supported by references of other works?
5	Are the contribution research clearly described?
6	Are there assumptions, if any, clearly stated?
7	Is there sufficient evidence to support the claims of the research?
	Experimental Evaluation
8	Is the research clearly described?
9	Is prototype, simulation or empirical study presented?
10	Is the experimental set up clearly described?
11	Are the results from multiple different experiments included?
12	Are the results from multiple runs of each experiment included?
13	Are the experimental results compared with other approaches?
14	Are negative results if any presented?
15	Is the statistical significance of the results assessed?
16	Are the limitations clearly stated?
17	Are links between data, interpretation and conclusions clear?

TABLE 4: Presents checklist for evaluating value of the study.

Thereafter, results were likened and disagreement resolved through discussion among researchers. Any paper not meeting minimum quality requirements as described below, was excluded from the set of primary studies.

Table 4 presents the checklist of the study quality valuation. For each question in the checklist a three- level, numeric scale was used. The levels were True (2 points), partial (1 point) and false (0 point). If the study scored 8 points or less, it was discarded due to the lack of quality in relation to this study. The research documented the obtained score of each included/excluded study.

2.9 Data Extraction

In this phase, researchers gathered information associated with the research questions from studies. To extract data, a form was created with a test-retest process for the reliability and correctness of the selected data. The form is indicated in Table. 5.

S/NO	Test Data Extraction Field	Explanation	Address
General Details			
PS01	Study_ID	Unique id for each primary study (PS)	
Study Description			
PS02	Title	Full title for the selected PS	
PS03	Author	Author's name of PS	
PS04	Year	Year study was published	
PS05	Type of Publication	Type of paper: Journal, conference, synopsis, book chapter, thesis etc.	
PS06	Publisher	Publisher's name	
Study Coverage			
PS07	Objective	Study's main objective	R1, R2, R3, R3,R4,R5,R6
PS08	Technique	Which metaheuristic techniques employed for test case optimization?	R1
PS09	Parameters	Which parameters are used by metaheuristic techniques for test case optimization?	R2
PS10	UML model	Which UML model (s) employed by metaheuristic technique for TCG?	R3
PS11	Intermediate format	Which Intermediate format employed by metaheuristic technique for TCG?	R4
PS12	Database	Which database used to assessthe metaheuristic techniques' performance?	R5
PS13	Metrics	Which metrics are applied to assessthe metaheuristic techniques' performance?	R6

TABLE 5: Primary studies data extraction form.

2.10 Synthesis of the Extracted Data

The extracted data from the papers was examined to get a high-level view of the different aspects related to TCP and optimization. The papers were categorized and collective results were extracted. The results from this phase are presented and discussed in Section 3.

3. REVIEW RESULTS

3.1 Review Details

In this sub-section, each study was analyzed and summarized to identify metaheuristic techniques used to enhance test case, the parameters, UML models, intermediate format applied and how the techniques are evaluated. To answer study questions, studies were reviewed and corresponding information was recorded. Table 6 presents the comprehensive results. The first column is the author(s) of the study, the second column is the algorithm used, third column defines the parameters applied, fourth column presents the UML diagram used, the fifth column

gives the intermediate format used, and sixth column is the database applied to evaluate the technique, seventh column is the metrics used and eighth column presents the research gaps.

Author	Algorithm	Parameters	UML Model	Intermediate Format	Databases	Metrics	Research Gap
(Shirole & Kumar, 2010)	Genetic Algorithm (GA)	not specified	Sequence	Sequence graph	,Application for stack calculator, System for Stu-dent Course Enrollment	covered, message sequence coverage	Test cases generated enhances test coverage. However, lacks balance between exploration and exploitation, test data is produced from a single model
(Biswal, 2010)	GA	not specified	Activity and collaboration diagram	not specified	ATM cash withdraw	transition coverage , single	Guarantees the minimum presence of error, in the generated test case. However it was not evaluated, lack balance between exploration and exploitation.
(Sabharwal et al., 2010)	GA	not specified	Activity	Control flow graph	Credit card membership	fitness value ,Path coverage , not evaluated	Efficacy is improved by finding the critical path bands. However need to be evaluated, lacks balance between exploration and exploitation, test data are generated from a single diagram
(Gulia, 2012)	GA	not specified	State chart	State flow diagram	Driverless Train	Not specified	Generates optimized sequence. However need to be evaluated, lacks balance between exploration and exploitation, test paths are created from an individual model
(Jena & Swain, 2012)	GA	not specified	Sequence	message control flow graph	ATM withdraw system	fitness value , message sequence path coverage , single	Generates and optimize test cases. However need to be evaluated, lacks balance between exploration and exploitation, only one diagram used to produce data
(Ranjan et al., 2013)	Firefly Algorithm (FA)	Number of flies (population size), No of iteration, number of edges, cyclomatic complexity,	State chart	Control flow graph	Generating a test case	No of states, cyclomatic complexity, path	Generates optimal paths. However need to be evaluated. lacks balance between exploration and

		objective function, absorption coefficient, distance (r)					exploitation, test paths are created from only one model
(Kaur & Kaur, 2013)	GA	not specified	Sequence	Tree structure	not specified	Edge-Pair Coverage and path coverage.	Finds more faults and increase the effectiveness through mutation testing. However need to be evaluated, lacks balance between exploration and exploitation, a single model is employed to create test data.
(Sumalatha, 2013)	GA	not specified	Sequence	sequence graph	Deal cards scenario	Fitness value, Path coverage.	not automated and also, need to be evaluated, lacks balance between exploration and exploitation, test paths are produced from an individual diagram
(Dalal & Chhillar, 2013)	Bee Colony Optimization (BC) and Modified Genetic Algorithm (MGA)	not specified	Activity	Activity dependency graph	Card management system	Fitness score , Decision node coverage	Proposes a tool to generate test cases automatically. However need to be evaluated. lacks balance between exploration and exploitation, test data are created from a single model
(Kumar & Husain, 2013)	GA	Not specified	Activity and sequence	Not specified	ATM withdrawal, balance enquiry with receipt and PIN verification	Faults detection score, single	Helps to reduce effort of generating test cases. However it lacks balance between exploration and exploitation.
(Jena, Ajay Kumar, Swain, Santosh Kumar, Mohapatra, n.d.)	GA	not specified	Activity	Activity Flow graph	ATM cash Withdrawal System	activity coverage criteria,	Not automated and also, need to be evaluated. lacks balance between exploration and exploitation, an individual model is employed to produce test data
(Hoseini, 2014)	GA	not specified	Sequence	control flow graph	User authentication in ATM	Path coverage	Major paths are automatically created and least paths extracted with shortest probable length. Need to be evaluated, lacks balance between exploration and

							exploitation, test data are produced from an individual diagram
(Mahali, 2014)	GA	not specified	Activity	Activity graph	Shopping mall management system	fitness value , Path coverage	Identified an enhanced independent path however, Need to be evaluated, lacks balance between exploration and exploitation, single model is used to create test data
(Ara & Biswas, 2014)	Tabu search	Not specified	Activity	Control flow graph	Library Management System.	cyclomatic complexity,	Ability to detect faults however, Need to be evaluated, lacks balance between exploration and exploitation
(Khurana et al., 2015)	GA	not specified	Activity, sequence, use case	System graph	Online examination system.	Fitness value, Path coverage.	Covers maximum number of faults. However, need to be automated, need to be evaluated, lacks balance between exploration and exploitation.
(Rhmann et al., 2015)	GA	Population size, No. of generations , Selection method, Crossover method, Mutation method	Activity	Activity flow graph	Book issue process from	Fitness value , Decision node coverage ,	Test paths are prioritized which helps to reduce testing cost. Need to be evaluated, lacks balance between exploration and exploitation, test data are produced from only one model
(Mandal et al., 2015)	Intelligent Optimization Algorithm	not specified	Activity	Activity graph	Shopping mall management system	path coverage	Generates optimized test suite. However, need to be evaluated through comparative study, lacks balance between exploration and exploitation and only a single model applied to create test paths.
(Khurana & Chillar, 2015)	GA	Population size, No. of generations , Selection method, Crossover method, Mutation method	Sequence and state chart diagrams	System Graph	Online based Voting System	No of test cases ,Path coverage	Identify and optimize test data –not automated. However, not evaluated, lacks balance between exploration and exploitation.
(Pradyot et al., 2015)	BAT	Frequency, velocity, loudness, location,	State chart	XML	class management system,	transition coverage	Generates favorable test sequence. However, lacks balance between

		desirability and probability.			enrollment study, telephone system		exploration and exploitation, test data created a single model
(Moussa et al., 2016)	Enhanced Anti colony optimization (ACO)	not specified	Activity, state chart, use case	Activity graph, state chart graph, use case graph	ATM system	cyclomatic complexity, test case generation time , Path coverage	Generate test data automatically from different behavioral diagrams. However need to evaluated, lacks balance between exploration and exploitation.
(Arifiani, 2016)	ACO	not specified	State chart	Dependency Graph	simple function	Average Coverage (AC), Success Rate (SR), Average (convergence) Generation (AG), and Average Time (AT), Branch coverage	Verify and validate requirement specification. However need to evaluated, lacks balance between exploration and exploitation, test data are produced from a single state chart
(Sahoo et al., 2017)	CS	Population size, Maximum number of generations, Levy flight, Random number, Probability of occurrence (pa),gamma, beta and sigma	Activity	Activity graph	ATM Withdraw operation	Path coverage , No of iterations , Fitness value	Generates ideal test cases, lacks balance between exploration and exploitation, test data are created from only one model
(Ansari, 2017)	FA	Number of flies (population size), maximum no. of generations, Initial value attraction coefficient (B0), light absorption factor (Y), mutation factor (α), mutation	Sequence	Adjacency Matrix	Patient Registration System	cyclomatic complexity	Generates optimal test data that can be used to identify faults, lacks balance between exploration and exploitation
(Sahoo, Rajesh Ku, Kumar et al., 2017)	Particle swarm-Bee Colony	Population size, no. of generation, velocity, pbest, gbest, probability factor, C1and C2, weight factor and random number	Activity and sequence	System testing graph	ATM Withdraw operation	Path coverage , total test case and total iteration,	Generates automated enhanced test paths. However, lacks balance between exploration and exploitation.
(Panthi & Mohapat	ACO	Pheromone value (Ph) (population size),	Activity	Activity Interaction Graph	Make Call	Weight value, not	Produce ranked test scenarios. However, lacks balance

ra, 2017)		Heuristic value(H), Visited status (Vs), Probability set (P), Sum, Condition (CD! =DN)				evaluate d	between exploration and exploitation, test paths are produced from a single diagram
(Basa et al., 2018)	GA	Not specified	State chart	state transition graph	Student Registratio n for Seminar	fitness value	Lacks balance between exploration and exploitation, test data are created from an individual state chart and it is semi-automatic
(Hashim & Dawood , 2018)	FA	Not specified	State chart	State relationshi p graph	ATM system	Path coverage	Produce test data identify faults like state based interaction, sequence and scenario faults and transaction based condition faults. However, lacks balance between exploration and exploitation, single state chart is used to produce test data.
(Saha, 2018)	Moth flame optimizatio n (MFO) algorithm	Distance between month (Dp), shape of the spiral (b), arbitrary number (t), extreme number of flames (Population size), total number of iterations (I)	State chart diagram	State graph	Translator for Braille To Text, ATM transactio n, Applicatio n for Microwave oven	No of test paths generate d, test paths generatio n time, redundan cy in test paths (%)	Produce minimal number of test paths when number of states is large. Lacks balance between exploration and exploitation, an individual state chart model is applied to create test paths.
(Lusiana et al., 2019)	GA	Not specified	Sequen ce and activity	System graph	ATM	Fitness value	Able to optimize generated test paths. However, needs to be evaluated, lacks balance between exploration and exploitation.
(Samah et al., 2019)	GA	Not specified	use case	use case	House Recomme ndation System	Fitness value	lacks balance between exploration and exploitation, test data are created from a single model
(Alrawas hed et al., 2019)	GA	Not specified	Use case	Control flow graph	File transfer protocol, ATM cash withdraw, Virtual meeting	transition coverage , no of generate d test cases, cyclomati c complexit y	Produce optimal test case However, lacks balance between exploration and exploitation, test data are produced from a single use case

(Rhman n, 2019)	FA	Absorption coefficient (Y),files' distance (rij),probability (j), generation no (t), random vector (ei), randomization (a), brightness of firefly (A0)	Activity	Control flow graph	Flight check-in process	cyclomati c complexit y, informati on flow	Generated optimized paths have no redundancy. lacks balance between exploration and exploitation, test data are created from a single model
(Panda & Dash, 2019)	Simulated Annealing-Cuckoo Search	Not specified	Sequen ce and state chart diagram	System graph	bench mark triangle classificati on problem	Transitio n path coverage	Generate test suits for transition path coverage and converges faster than Cuckoo Search and SA algorithms.
(Rastogi , 2019)	Grey Wolf-Firefly	Population size, no of iterations, alpha, beta, delta, Absorption coefficient (Y),files' distance (rij), random vector (ei), randomization (a), brightness of firefly (A0)	State chart, sequenc e	State chart sequence diagram graph	ATM	Path coverage , Fitness value, no of iterations , mean time between failures (MTBF), execution time	Achieves a higher appropriate objective value. However, lacks balance between exploration and exploitation
(Jaffari et al., 2020)	GA	Population size, No. of generations , Selection method, Crossover method, Mutation method	Activity	XML	Cruise control, coffee maker, elevator	Statemen t and branch coverage	Capable of discovering faults. However, lacks balance between exploration and exploitation, test paths are created from an individual model
(Panda et al., 2020)	Firefly-Differential Evolution (FA-DE)	first value attraction factor (B0), light absorption factor (Y), mutation factor (α), mutation factor damping ratio (alpha_damp), selection rate (R), scaling factor for mutation (F) and crossover rate (Pc).	State chart	Statechart diagram graph	Triangle classificati on	Transitio n path coverage , Total No.of test case, total execution time. mean number of test case, minimum no of test cases	Produce enhanced test case. However, lacks balance between exploration and exploitation, test data are created from a single model.
(Sankar & Chandra , 2020)	ACO	Heuristic value(H), Pheromone Intensity Visited status (Vs), Probability (P), alpha, beta	State chart	State graph	triangle problem, quadratic equation problem	Statemen t Branch, Decision Coverag e, Average % of	Produces optimal test cases that ensure maximum coverage. However, lacks balance between exploration and exploitation, test

						Faults Discovered (APFD) and % of Test Case Required (PTR) metric	data are produced from an individual diagram
(Lakshminarayana & Sureshkumar, 2020)	CS-BC	Control parameter, No. of iteration, Levy (searching vector), Candidate solution, Random number, Probability of occurrence (pa)	State chart & Sequence	SCSEDG graph	ATM, Soft Drink Selling Machine	Fitness value, Total Iterations, No of Test cases, Transition Coverage, execution time ,Mean Time Between Failures	Takes less time for the production of path coverage. And achieved 65% of test data with a higher fitness function value. However, there is need to improve the efficiency of the technique, balance between exploration & exploitation
(Panigrahi et al., 2021)	FA-BC	Number of generations, Random value {-1 to +1}, Probability of occurrence (pa), Vmin (Lowest balance), Attractiveness, Beta, Gamma, Alpha.	Activity diagram	Activity Dependency Flow Graph	ATM	Fitness value, No of Iterations, No of Test cases , Path Coverage	Optimal solution is achieved after 90 iterations. Generates test data from cases from a single diagram
(Sahoo, Satpathy, et al., 2021)	ACSA	Population size, Maximum number of generations, Levy flight, Random number, Probability of occurrence (pa)	Sequence	Sequence diagram graph	ATM withdraw operation	Fitness value, No of Iterations Path Coverage	Optimal solution is achieved after 100 iterations. However, it Lack balance between exploration & exploitation and generates test data from a single diagram
(Sahoo, Derbali, Jerbi, van Thang, et al., 2021)	GA	Population size, Initial best solution, Fitness function value, No. of generations, Selection method, Crossover method, Mutation method	Activity & state chart	Activity StateChart Graph	An operation for ATM withdraw	Total Iterations, No of Test cases	Produced optimal result after 160th iterations. However, Lack balance between exploration & exploitation
(Tamizharasi, A., Ezhumal	BFA-PSO-GA	Initial position, step size C, velocity, maximum value,	Activity	Activity diagram graph	Online shopping	path coverage , Executio	The total test data produced is 60 with a total time of 29.3 sec. However, Lack

ai, P., Remya Rose, S. , Sureshd , P., Logess warie, 2021)		C1and C2 (Cognition and social components), pbest, gbest, selection and crossover techniques				n time, No. of test paths, iteration taken,	balance between exploration & exploitation
(Tamizh arasi & Ezhumal ai, 2022)	Genetic-based Crow Search	population size, awareness likelihood, extreme cycle limit, crow original position, random number, selection method, crossover method and mutation method	Activity diagram	Control Flow Graph	Fund transfer in Net Banking Applicatio n	Cyclomat ic Complexi ty, Fitness value, Total test case, Total time ,Total Path Covered	Able to covers 100% of paths with less execution time. However Lack balance between exploration & exploitation and creates test data from an individual activity diagram
(Raame sh & Jothi, 2022)	shuffled shepherd flamingo search (S2FS)	Maximum no of iterations, size of population, population renewal, diffusion factor, initial member (δ), random vector, step size (α)	State chart	State chart graph	ATM withdraw operation	Average time among failures, Total time taken, no of test case, Total iterations	Able to achieve a higher aptness value. However creates test data from an individual State chart
(Potluri et al., 2022)	PS-BC, FA-CS	Fitness function, population size , particle swarm population size, PSO parameters (k1,k2), PSO weight factor, no of iteration bee colony population size, Fire fly population size, cuckoo search population size, beta attractiveness	State Chart ,Sequen ce	State Chart Sequence Diagram Graph	ATM transactio n	Aptness value, Total case, Total iterations	Able to produce better results after 160th generations
(Tatale & Prakash , 2022)	PSO	Initial no.of particles, no. of iterations, velocity, weight factor (w), acceleration coefficient (c1and c2) and random values (r1 and r2), dimension (d)	Activity	XML	Railway Reservatio n System	No. of test case generate d, Accuracy	An overall of 75 test data were created. However, However Lack balance between exploration & exploitation and generates test cases from a single diagram

TABLE 6: Summary of Review Details.

3.2 Analysis

The section consist examination of work studied and responses of the considered queries for the research. After reviewing the 46 major studies and categorizing the associated data, the subsequent investigation is employed to respond each research query.

3.2.1 Analysis for Research Question 1

How are metaheuristic techniques employed for test case optimization?

The principle behind RQ1 was to recognize metaheuristic algorithm applied by the primary studies to create and improve test paths or test case. Figure 4 shows that greatest number of studies (20) utilized GA, to produce and enhance test case, while 4 FA and ACO had each been applied by 4 studies. The CS was only used by 2 studies while BAT, MFO, IOA, GA-CS,FA-DE, FA-BC,CS-BC,S2FS,PS-BC & FA-CS,BFA-PSO-GA, PSO-BC, GW-MGA, SA-CS, GW-FA, BC-MGA,GA-ACO and Tabu search had each been applied by one study.

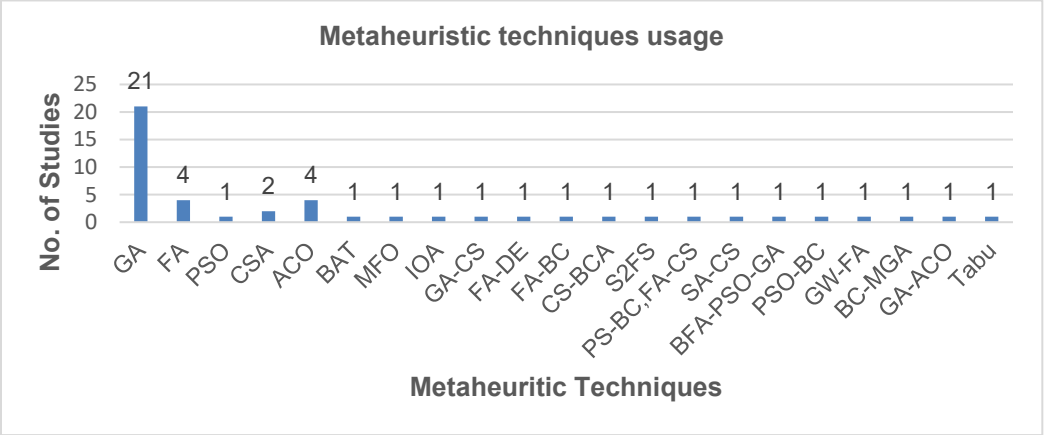

FIGURE 4: Optimization Techniques.

3.2.2 Analysis for Research Question 2

Which are the common parameters used by metaheuristic techniques for test case optimization?

The RQ2 focused on identifying the parameters used by metaheuristic techniques for optimization of test cases. The findings from table 6 show that every metaheuristic algorithm has its own specific parameters. However, there are parameters that are common to all algorithms such as the population size which define the total test case, total iterations which defines the highest total iterations an algorithm may executes, random value, probability of occurrence, beta, gamma, alpha and sigma which are used as the scaling factors.

Figure 5 indicates that the number of generations and population size parameters were used by 78% of studies, 43% of studies applied random value parameter, probability of occurance parameter was used by 39% of studies, the sigma parameter was applied by 13% of studies while alpha and beta parameters had each used by 22% of the studies. Conversely, 50% of the studies were unable not specify parameters.

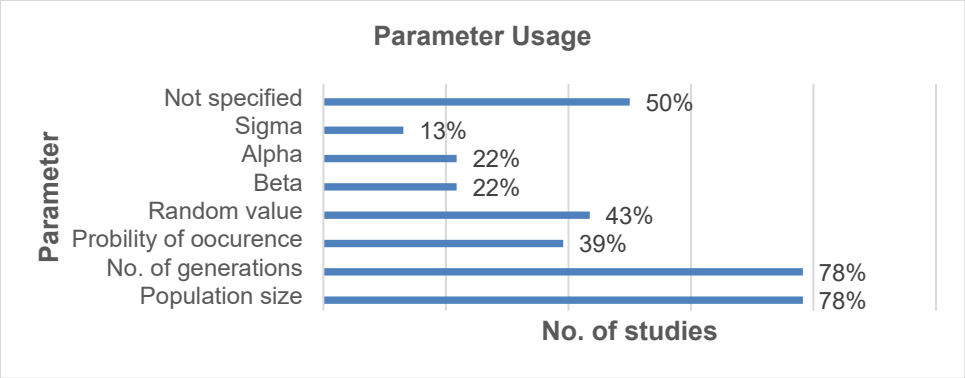

FIGURE 5: Parameters usage.

3.2.3 Analysis for Research Question 3

How are UML models applied by metaheuristic techniques for TCP?

The question RQ3's idea was to find the UML models used for creating test case. The findings indicates that indicates that 15 primary studies generated test cases from activity diagram, 10 studies used state chart diagram to generate, 7 primary studies used sequence diagram, and use case diagram was used by 3 primary studies while 11 primary studies used combinational UML diagrams.

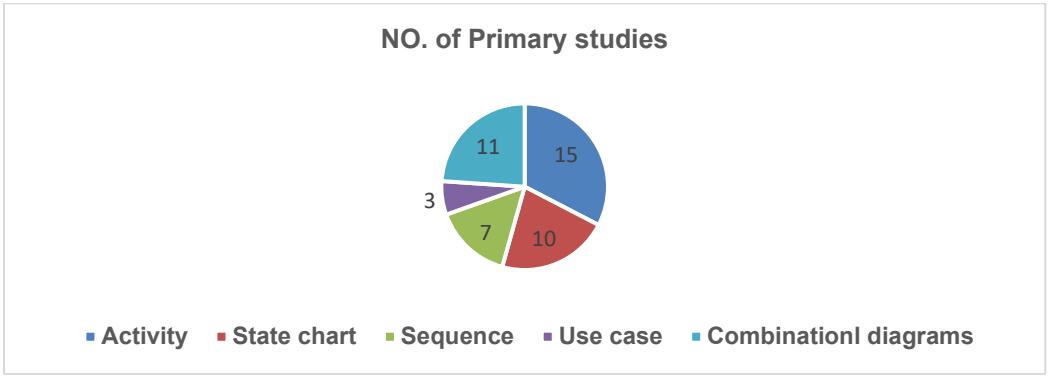

FIGURE 6: UML models employed by metaheuristic techniques.

3.2.4 Analysis for Research Question 4

Which intermediate format used by metaheuristic techniques for TCP?

The focus of this question (RQ4) was to determine the intermediate format that techniques uses to create test paths. From the findings, it indicates that 83% of studies used graph as the intermediate formats while 7% employed XML code, 2% of studies applied adjacency matrix while 4% of the studies did not specify the intermediate format

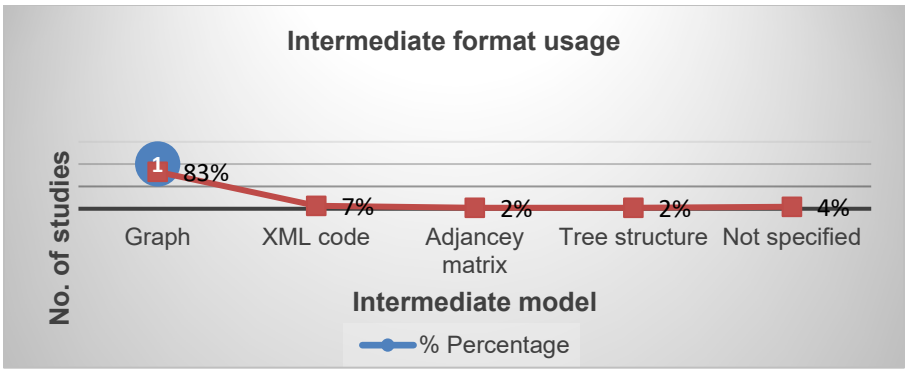

FIGURE 7: Intermediate format used by metaheuristic techniques.

3.2.5 Analysis for Research Question 5

Which databases are employed to evaluate the performance of metaheuristic techniques?

The question RQ5 aimed at determining databases applied to evaluate the metaheuristic techniques. Figure 8 shows that ATM was the most used database by the studies to evaluate metaheuristic techniques. The ATM database was used by 21 studies out of 46 studies, while 18 studies out of 46 studies applied simple programs. Credit card management was used by 2 studies. The database that included soft drink vending machine, telephone system, online examination system, patient registration system, online voting system, shopping management system, driverless train and library management system had each applied by one study. One study does not specify its database.

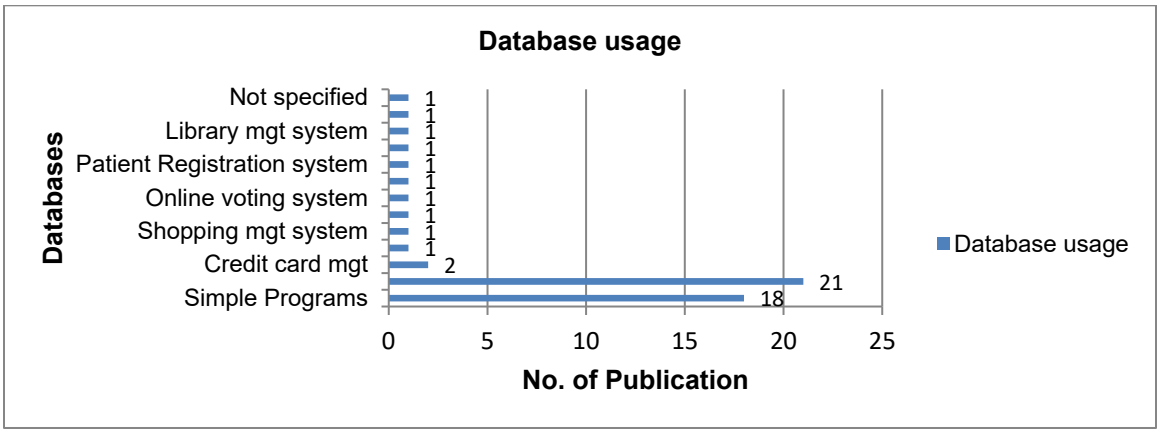

FIGURE 8: Databases used to evaluate the metaheuristic techniques.

3.2.6 Analysis for Research Question 6

Which metrics are used to evaluate the performance of metaheuristic techniques?

The aim of this question (RQ6) was to identify the metrics used to assess how efficiently and effectively test has been performed. Figure 9 shows path coverage and fitness were used by 21 and 17 primary studies respectively whereas message path coverage metric was employed by 11 studies. No. of test case and No. of iterations were each applied by 10 studies whereas execution time and transition coverage metrics were each used by 8 studies. Node coverage and MTBF metrics were used by 3 studies whereas fault detection score metric was used by 2 studies. Metrics that include; Edge pair coverage, APFD, AC, AT, Accuracy and No. of states had each employed by 1 study only.

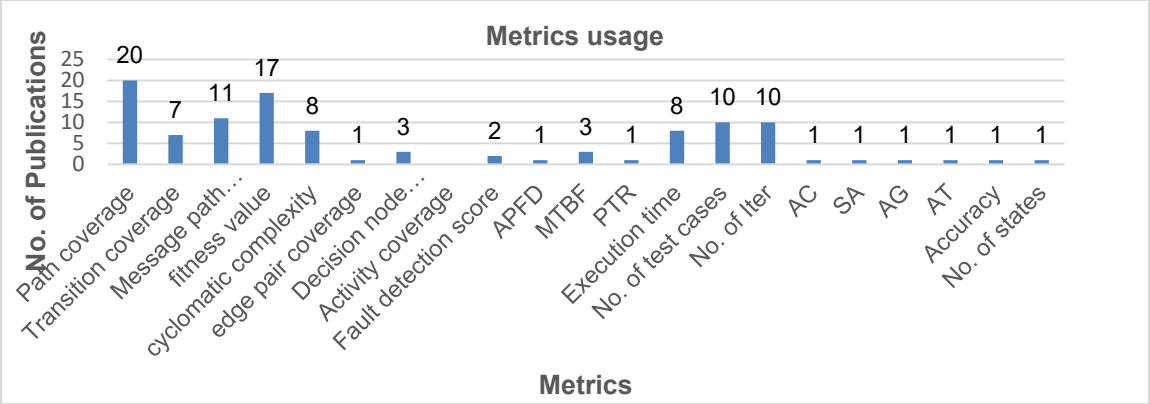

FIGURE 9: Metrics applied to assess performance of metaheuristic performance.

Table 7 is a matrix that highlights the techniques, parameters, models, formats, databases, and evaluation metrics employed in metaheuristic-based test case optimization. In addition to summarizing the findings, the matrix identifies gaps and observations that expose current research limitations and point toward potential future directions.

Research Question	Aspect Analyzed	Key Findings	Gaps/Observations
RQ1: Metaheuristic Techniques	Algorithms used for test case optimization	Genetic Algorithm (GA) used by 20 studies; FA and ACO by 4 each; CS by 2; others (BAT, MFO, S2FS, hybrids) by 1 each	Over-reliance on GA; limited use of newer or hybrid algorithms
RQ2: Parameters Used	Common parameters in metaheuristic optimization	Population size, number of iterations used by 78% of studies; random value (43%), probability of occurrence (39%), alpha & beta (22%), sigma (13%); 50% did not specify parameters	Static parameter use limits adaptability; poor documentation reduces reproducibility
RQ3: UML Models	UML models for test case generation	Activity diagram (15 studies), State chart (10), Sequence diagram (7), Use case (3), Combinations (11)	Heavy reliance on single models; limited integration of multiple diagrams
RQ4: Intermediate Formats	Formats for representing test paths	Graphs used by 83%; XML (7%), adjacency matrix (2%), unspecified (4%)	Limited exploration of scalable, automation-friendly formats like XML, XMI
RQ5: Databases for Evaluation	Databases used for validation	ATM systems used by 21 studies; simple programs (18), other systems (1 each), 1 study unspecified	Narrow range of test systems; limited generalizability
RQ6: Evaluation Metrics	Metrics used to assess metaheuristic performance	Path coverage (21 studies), fitness (17), message path coverage (11), number of test cases & iterations (10), execution time & transition coverage (8), others rarely used	Limited diversity in metrics; need for broader evaluation criteria including execution time, fault detection

TABLE 7: Comparison matrix for the results.

4. DISCUSSION

This section provides evaluation and research gaps between associated research studies on parameters, UML models, intermediate formats, evaluation metrics applied by metaheuristic techniques in order to produce and improve test case.

Regarding the metaheuristic techniques applied for test case optimization, the analysis reveals that Genetic Algorithm (GA) is the most frequently used in primary studies, likely due to its simplicity and adaptability. However, GA often suffers from high computational cost due to a large number of generations and the risk of premature convergence. This signals the need for broader experimentation with newer or hybrid algorithms, which remain largely underutilized in the existing literature.

Expanding on this, the analysis of commonly used parameters in metaheuristic techniques reveals that variables such as population size, number of iterations, random values, probability of occurrence, and scaling factors (alpha, beta, sigma) are frequently applied in test case optimization. However, these parameters are often applied statically, which contributes to an imbalance between exploration and exploitation. Additionally, half of the reviewed studies did not report parameter settings, and those that did often lacked sufficient documentation, limiting the reliability and reproducibility of existing literature.

The findings also reveal an over-dependence on single UML models, such as activity, state chart or sequence diagrams, when generating test cases. This practice may oversimplify the testing process and overlook critical system behaviors. A more comprehensive strategy involving the integration of multiple UML diagrams could improve test coverage and provide a richer understanding of system interactions.

Moreover, most studies favor graph-based intermediate representations for generating test paths. While effective in some scenarios, these formats may pose scalability challenges and limit automation capabilities, especially in complex systems. There is significant potential in exploring alternative representations, including XML, XMI, adjacency matrices, and spreadsheet-based models, which could offer greater flexibility and automation support.

A further limitation lies in the narrow focus of database selection for validation purposes. Many studies predominantly utilize ATM system scenarios, which raises concerns about the generalizability of their findings to other real-world applications. Future research should prioritize more diverse and complex datasets to comprehensively evaluate the performance of test case generation techniques.

Additionally, the heavy reliance on a limited set of evaluation metrics primarily path coverage and fitness value fails to fully capture the performance spectrum of metaheuristic approaches. Broader evaluation frameworks incorporating metrics such as execution time, fault detection capability, and coverage diversity could provide a more balanced and informative assessment.

In summary, while substantial progress has been made in applying metaheuristic algorithms to software test case generation, critical gaps remain. Addressing these gaps through the adoption of innovative algorithms, dynamic parameter tuning, integrated UML modeling, advanced intermediate formats, diversified validation environments, and comprehensive evaluation metrics could significantly enhance the efficiency, scalability, and effectiveness of future software testing solutions

5. THREATS TO VALIDITY

The first threat relates to search strings. The procedure of building search strings generallydepends on past experience to define the content of strings. Even though we constructed the search strings carefully and performed the automatic search on the relevant databases, there is no guarantee that we find and select all possible research approaches. For example, the works published as internal technical reports, company journals or written in other languages are not available for study. In this literature review, the neglected works may have crucial contributions and affect the completeness of this review. To address this threat, the process of creating inclusive search strings and appropriate exclusion criteria needs to be continually reviewed. In the step of confirming primary studies we read each approach and

distinguished the works carefully, to ensure that they met the criteria.The second threat to validity is that, only research papers from four databases i.e., IEEE Explorer, Springer, Elsevier and Google scholar were included. Some relevant papers from other databases may have been left out. However, the use of google scholar minimized the threat since it was able to link to papers in other databases such as ACM.

The last threat to validity is that the screening phases were performed partially by different persons. While one researcher followed the entire protocol from beginning to end, the remaining researcher had varying influence on the screening phases. These researchers may have had different views regarding paper relevancy, causing relevant papers to be excluded. In all phases where two researchers were involved, except for the data extraction phase, one researcher completed the entire phase independently, while the other two divided the workload evenly between them. Since the workload was divided, some papers may have been excluded because of differing criteria for relevance. In the data extraction phase, each of the researchers extracted data from one third of the papers. Although each set of extracted data was double-checked by other researcher, there is a risk that some case may have been missed. Finally, the researchers pointed out that after each phase in the protocol, consensus discussions were held and that any disagreements were resolved. Therefore, the researchers feel that any threats posed to protocol execution were minimized.

6. CONCLUSION AND FUTURE WORK

This study provides a comprehensive review of metaheuristic techniques used for test case optimization, guided by specific research questions (Section 2.3). A total of 46 primary studies were selected based on quality assessment criteria (Section 2.8). The findings reveal that both hybrid and single metaheuristic approaches are utilized (RQ1), though the majority of studies favored single techniques, particularly Genetic Algorithms. Regarding RQ2, half of the studies did not specify the parameters used for optimization, and those that did often overlooked the critical balance between exploration and exploitation, which affects algorithm efficiency. For RQ3, most research focused on individual UML diagrams, especially activity diagrams. Concerning RQ4, graphs were the predominant intermediate format for generating test cases, with limited use of XML. In relation to RQ5, the ATM system emerged as the most frequently used database for performance evaluation. Lastly, for RQ6, most studies prioritized effectiveness metrics, while only a few considered efficiency-related measures.

The results of this review carry significant academic and practical value. From an academic perspective, the study enriches existing knowledge by highlighting key gaps, including insufficient parameter documentation, limited adoption of hybrid techniques, and a restricted emphasis on UML models and evaluation metrics. These gaps open avenues for future research aimed at developing more advanced, efficient, and well-documented optimization methods. On the practical side, the findings offer useful guidance for software engineers, testers, tool developers, and quality assurance professionals in improving test case generation by enhancing parameter selection, balancing exploration and exploitation, and integrating multiple models for more effective and cost-efficient testing.

In the near future, an expert opinion survey will be conducted to identify key parameters for generating and improving test cases, addressing the current gap in parameter documentation and empirical validation in existing studies. Based on these findings, an efficient technique will be developed that maintains a balance between exploration and exploitation factors and supports the generation of test paths from multiple UML diagrams. To evaluate the proposed approach, a comparative study using large datasets will be performed, with particular focus on both efficiency and effectiveness metrics.

7. REFERENCES

Abayatilake, P., & Blessing, L. (2021). The Application of Function Models In Software Design: A Survey Within the Software Community. I*nternational Journal of Software Engineering,* 9(9), 27–62. https://www.cscjournals.org/library/manuscriptinfo.php?mc=IJSE-176

Aditi, Park, H., Sung, S., Han, Y.-S., & Ko, S.-K. (2025). *SAGE:Specification-Aware Grammar* Extraction for Automated Test Case Generation with LLMs. http://arxiv.org/abs/2506.11081

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

Alzaqebah, M., Briki, K., Alrefai, N., Brini, S., Jawarneh, S., Alsmadi, M. K., Mohammad, R. M. A., ALmarashdeh, I., Alghamdi, F. A., Aldhafferi, N., & Alqahtani, A. (2021). Memory based cuckoo search algorithm for feature selection of gene expression dataset. I*nformatics in Medicine* Unlocked, 24, 100572. https://doi.org/10.1016/j.imu.2021.100572

Ansari, G. A. (2017). Use of Firefly Algorithm in Optimization and Prioritization of Test Paths Generated from UML Sequence Diagram. 167(4), 24–30.

Ara, M., & Biswas, H. A. (2014). A Novel Approach for Test Path Generation and Prioritization of UML Activity Diagrams using Tabu Search Algorithm. *International Journal of Scientific & Engineering Research*, 5(2), 1212–1217.

Arifiani, S. (2016). Generating Test Data Using Ant Colony Optimization (ACO) Algorithm and UML State Machine Diagram in Gray Box Testing Approach. *2016 International Seminar on Application for Technology of Information and Communication (ISemantic)*, 217–222. https://doi.org/10.1109/ISEMANTIC.2016.7873841.

Basa, S. S., Swain, S. K., & Mohapatra, D. P. (2018). Genetic Algorithm-based Optimized Test Case Design Using UML Genetic Algorithm-based Optimized Test Case Design Using UML. September. https://doi.org/10.29055/jcms/862.

Biswal, B. N. (2010). A Novel Approach for Optimized Test Case Generation Using Activity and Collaboration Diagram. 1(14).

Cuong-le, T., Hoang-le, M., Khatir, S., Wahab, M. A., Tran, M. T., & Mirjalili, S. (2021). A novel version of Cuckoo search Algorithm for solving Optimization problems. *Expert Systems With Applications,* 115669. https://doi.org/10.1016/j.eswa.2021.115669.

Dalal, S., & Chhillar, R. S. (2013). A Novel Technique for Generation of Test Cases Based on Bee Colony Optimization and Modified Genetic Algorithm. 68(19).

Fan, L., Wang, Y., & Liu, T. (2021). Automatic Test Path Generation and Prioritization using UML Activity Diagram. 484–490. https://doi.org/10.1109/dsa52907.2021.00072.

Gulia, P. (2012). New Approach to Generate and Optimize Test Cases for UML State Diagram Using Genetic Algorithm Categories and Subject Descriptors : General Terms : *ACM SIGSOFT Software Engineering Notes*. 37(3), 2–6. https://doi.org/10.1145/180921.2180933

Hasan, N. Bin, Islam, M. A., Khan, J. Y., Senjik, S., & Iqbal, A. (2025). Automatic High-Level Test Case Generation using Large Language Models. *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, 674–685. https://doi.org/10.1109/MSR66628.2025.00105.

Hashim, N. L., & Dawood, Y. S. (2018). Test case minimization applying firefly algorithm. *International Journal on Advanced Science, Engineering and Information Technology*, 8(4–2), 1777–1783. https://doi.org/10.18517/ijaseit.8.4-2.6820.

Hoseini, B. (2014). Automatic Test Path Generation from Sequence Diagram Using Genetic Algorithm. 106–111.

Jaffari, A., Yoo, C. J., & Lee, J. (2020). Automatic test data generation using the activity diagram and search-based technique. Applied Sciences (Switzerland), 10(10), 9–13. https://doi.org/10.3390/APP10103397.

Jena, Ajay Kumar, Swain, Santosh Kumar, Mohapatra, D. P. (n.d.). A Novel Approach for Test Case Generation from UML Activity Diagram. *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT),* 621–629. https://doi.org/10.1109/ICICICT.2014.6781352.

Jena, A. K., & Swain, S. K. (2012). Test Case Creation from UML Sequence Diagram : A Soft Computing Approach. https://doi.org/10.1007/978-81-322-2012-1.

Kaur, P., & Kaur, R. (2013).Approaches for Generating Test Cases Automatically to Test the Software. *International Journal of Engineering and Advanced Technology (IJEAT)*, 3, 2249–8958.

Khurana, N., Chhillar, R. S., & Chhillar, U. (2015). A Novel Technique for Generation and Optimization of Test Cases Using Use Case , Sequence , Activity Diagram and Genetic Algorithm. 11(3), 242–250. https://doi.org/10.17706/jsw.11.3.242-250.

Khurana, N., & Chillar, R. S. (2015). Test Case Generation and Optimization using UML Models and Genetic Algorithm. *Procedia Computer Science*, 57, 996–1004. https://doi.org/10.1016/j.procs.2015.07.502.

Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O. P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering-A tertiary study. *Information and Software Technology,* 52(8), 792–805. https://doi.org/10.1016/j.infsof.2010.03.006.

Kumar, M., & Husain, P. M. (2013). Test Cases Optimization Evaluation Using Efficient Algorithm with UML. 1, 16–20.

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.

Li, J., Xiao, D., Lei, H., Zhang, T., & Tian, T. (2020). Using Cuckoo Search Algorithm with Q - Learning and Genetic Operation to Solve the Problem of Logistics Distribution Center Location.

Lusiana, M., Dewi, C., & Chandra, A. (2019). Optimization of test case generation from uml Activity diagram and sequence diagram By using genetic algorithm. *ICIC Express Letters*, 13(7), 585–591. https://doi.org/10.24507/icicel.13.07.585.

Mahali, P. (2014). Model Based Test Case Prioritization Using UML Activity Diagram and Evolutionary Algorithm. I*nternational Journal of Computer Science and Informatics* Volume, 4(2). https://doi.org/10.47893/IJCSI.2014.1177

Mandal, J. K., Satapathy, S. C., Sanyal, M. K., Sarkar, P. P., & Mukhopadhyay, A. (2015). Information systems design and intelligent applications: Proceedings of second international conference India 2015, volume 1. *Advances in Intelligent Systems and Computing*, 339. https://doi.org/10.1007/978-81-322-2250-7.

Mburu, J. M., Muketha, G. M., & Oirere, A. M. (2020). An Enhanced Multiview Test Case Generation Technique for Object-oriented Software using Class and Activity Diagrams. 4, 186–195. https://doi.org/10.35940/ijrte.D4908.119420.

Mburu, J. M., & Ndia, J. G. (2022). A Systematic Mapping Study on UML Model based Test Case Generation and Optimization Techniques. 184(13), 26–33.

Mohd-Shafie, M. L., Kadir, W. M. N. W., Lichter, H., Khatibsyarbini, M., & Isa, M. A. (2022). Model-based test case generation and prioritization: a systematic literature review.*In Software and Systems Modeling* (Vol. 21, Issue 2). https://doi.org/10.1007/s10270-021-00924-8.

Moussa, S., Elghondakly, R., & Badr, N. (2016). An Optimized Approach for Automated Test Case Generation and Validation for UML diagrams. September. https://doi.org/10.3923/ajit.2016.4276.4290.

Panda, M., & Dash, S. (2019). A Framework for Testing Object Oriented Programs Using Hybrid Nature Inspired Algorithms. *Springer* Singapore. https://doi.org/10.1007/978-981-13-3140-4.

Panda, M., Dash, S., Nayyar, A., Bilal, M., & Mehmood, R. M. (2020). Test suit generation for object oriented programs: A hybrid firefly and differential evolution approach. *IEEE Access,* 8, 179167–179188. https://doi.org/10.1109/ACCESS.2020.3026911.

Panigrahi, S. S., Sahoo, P. K., Sahu, B. P., Panigrahi, A., & Jena, A. K. (2021). Model-driven automatic paths generation and test case optimization using hybrid FA-BC. *2021 International Conference on Emerging Smart Computing and Informatics, ESCI* 2021, 263–268. https://doi.org/10.1109/ESCI50559.2021.9396999.

Panthi, V., & Mohapatra, D. P. (2017). ACO based embedded system testing using UML Activity Diagram. *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 237–242. https://doi.org/10.1109/TENCON.2016.7847997.

Potluri, S., Ravindra, J., Mohammad, G. B., & Sajja, G. S. (2022). Optimized Test Coverage with Hybrid Particle Swarm Bee Colony and Firefly Cuckoo Search Algorithms in Model Based Software Testing.*IEEE*.

Pradyot, K., Sharma, D., & Gouthami, K. P. (2015). Favourable test sequence generation in state-based testing using bat algorithm https://doi.org/10.1504/IJCAT.2015.070495.

Raamesh, L., & Jothi, S. R. S. (2022). Generating Optimal Test Case Generation Using Shuffled Shepherd Flamingo Search Model. N*eural Processing Letters*. https://doi.org/10.1007/s11063-022-10867-w.

Ranjan, P., Mallikarjun, B., & Yang, X. (2013). Optimal test sequence generation using firefly algorithm. 8, 44–53.

Rao, C. P. (2016). Comprehensive Testing Tool for Automatic Test Suite Generation , Prioritization and Testing of Object Oriented Software Products. *International Journal of Software Engineering*, 7(1), 1–15.

Rastogi, P. (2019). An Optimal Software Test Case Mechanism using Grey Wolf-FireFly Method. 12(2), 22–32. https://doi.org/10.22266/ijies2019.0430.03.

Rhmann, W. (2019). Optimized and Prioritized Test Paths Generation from UML Activity Diagram Optimized and Prioritized Test Paths Generation from UML Activity Diagram using Firefly Algorithm. June. https://doi.org/10.5120/ijca2016910718.

Rhmann, W., Zaidi, T., & Saxena, V. (2015). Test Case Generation and Optimization using UML Models and Genetic Algorithm. *International Journal of Computer Applications,* 115(4), 8–12. https://doi.org/10.5120/20137-2232.

Sabharwal, S., Sibal, R., & Sharma, C. (2010). Prioritization Of Test Case Scenarios Derived From Activity Diagram Using Genetic Algorithm. 2010 *International Conference on Computer and Communication Technology (ICCCT)*, 481–485. https://doi.org/10.1109/ICCCT.2010.5640479.

Saha, R. S. and A. (2018). Optimal test sequence generation in state 2 based testing using moth flame optimization 3 algorithm. https://doi.org/10.3233/JIFS-169804.

Sahoo, Rajesh Ku, Kumar, S. N., Mohapatra, D. P., & Patra, M. R. (2017). Model Driven Test Case Optimization of UML Combinational Diagrams Using Hybrid Bee Colony Algorithm. June, 43–54. https://doi.org/10.5815/ijisa.2017.06.05.

Sahoo, R. K., Derbali, M., Jerbi, H., van Thang, D., Kumar, P. P., & Sahoo, S. (2021). Test Case Generation from UML-Diagrams Using Genetic Algorithm. 67(2), 2321–2336. https://doi.org/10.32604/cmc.2021.013014.

Sahoo, R. K., Mohapatra, D. P., & Patra, M. R. (2017). Model Driven Approach for Test Data Optimization Using Activity Diagram Based on Cuckoo Search Algorithm. *International Journal of Information Technology and Computer Science,* 9(10), 77–84. https://doi.org/10.5815/ijitcs.2017.10.08.

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*. https://doi.org/10.1007/s11334-020-00378-z.

Samah, K. A. F. A., Badarudin, I. M., Odzaly, E. E., Ismail, K. N., Nasarudin, N. I. S., Tahar, N. F., & Khairuddin, M. H. (2019). Optimization of house purchase recommendation system (HPRS) using genetic algorithm. *Indonesian Journal of Electrical Engineering and Computer Science*, 16(3), 1530–1538. https://doi.org/10.11591/ijeecs.v16.i3.pp1530-1538.

Sankar, S., & Chandra, V. (2020). An Ant Colony Optimization Algorithm Based Automated Generation of Software Test Cases (Vol. 1). *Springer International Publishing.* https://doi.org/10.1007/978-3-030-53956-6.

Shirole, M., & Kumar, R. (2010). A hybrid genetic algorithm based test case generation using sequence diagrams. *Communications in Computer and Information Science*, 94 CCIS(PART 1), 53–63. https://doi.org/10.1007/978-3-642-14834-7_6.

Sumalatha, V. M. (2013). Object Oriented Test Case Generation Technique using Genetic Algorithms. 61(20), 20–26.

Tamizharasi, A., Ezhumalai, P., Remya Rose, S. , Sureshd, P., Logesswarie, S. (2021). Bio Inspired Approach for Generating Test data from User Stories. *Turkish Journal of Computer and Mathematics Education (TURCOMAT),* 12(2), 412–419. https://doi.org/10.17762/turcomat.v12i2.826.

Tamizharasi, A., & Ezhumalai, P. (2022). Genetic-based Crow Search Algorithm for Test Case Generation. 1–11. https://doi.org/10.14456/ITJEMAST.2022.74.

Tatale, S., & Prakash, V. C. (2022). Ingénierie des Systèmes d ' Information Automatic Generation and Optimization of Combinatorial Test Cases from UML Activity Diagram Using Particle Swarm Optimization. 27(1), 49–59.

Wambui, A., Muketha, G. M., & Ndia, J. G. (n.d.). A Framework for Analyzing UML Behavioral Metrics based on Complexity Perspectives. 11, 1–12.

Xiong, Y., Zou, Z., & Cheng, J. (2023). Cuckoo search algorithm based on cloud model and its application. *Scientific Reports*, 13(1), 1–13. https://doi.org/10.1038/s41598-023-37326-3.