Metaheuristic-Based Test Case Selection for Regression Testing: A Systematic Literature Review

Siti Hawa Mohamed Shareef[1], Rabatul Aduni Sulaiman[2]*, Nazri Mohd Nawi[3],
Wan Noor Hamiza Wan Ali[4], Nurul N. Jamal[5], Fairuz Amalina[6]
Department of Software Engineering-Faculty of Computer Science and Information Technology,
University Tun Hussein Onn Malaysia, Batu Pahat, Malaysia[1, 2, 3]
Department of Intelligence Informatics-Faculty of Artificial Intelligence,
University Technology Malaysia, Kuala Lumpur, Malaysia[4]
Information System Group-Faculty of Computer Science and Mathematics,
University Malaysia Terengganu, Kuala Terengganu, Malaysia[5]
Department of Information System-Faculty of Computer Science and Information Technology,
Universiti Malaya, Kuala Lumpur, Malaysia[6]

***Abstract*—Regression testing plays a crucial role in ensuring that software modifications do not adversely affect existing functionality. However, test suites continue to grow, and the retest-all method has become increasingly impractical due to high execution costs and time constraints. Consequently, Test Case Selection (TCS) has emerged as an important optimization method to identify which test cases are relevant for re-execution. Metaheuristic optimization has received a great deal of attention in Search-Based Software Testing (SBST) for balancing the conflicting objectives of cost reduction and fault detection effectiveness. However, there are still multiple areas of fragmentation within the research, including algorithm design, objective formulation, empirical evaluation, and reporting practices. Results indicate that evolutionary algorithms and swarm intelligence are the primary algorithms used for TCS, with an increasing trend towards hybrid and multi-objective approaches to increase the quality and scalability of the search. Multi-objective formulations are significant in this context, as such approaches provide a structured mechanism to capture the trade-off between testing cost and effectiveness through Pareto-based evaluations. Empirical evidence remains largely dependent on benchmark and open-source systems, although recent studies increasingly incorporate industrial and Artificial Intelligence (AI)-based environments. Overall, the findings indicate that there is a lack of single techniques that are universally superior, as performance is strongly influenced by system characteristics and evaluation criteria. Future research should emphasize standardized evaluation practices, stronger industry-scale validation, and the development of domain-aware TCS techniques to improve practical applicability.**

***Keywords*—Regression testing; test case selection; metaheuristic algorithms; search-based software testing**

I. Introduction

Regression testing as part of software quality assurance ensures that newly made changes do not affect previously functional parts of the application [1]. However, as the size of the test suite grows larger, running retest-all becomes extremely time-consuming [2]. To improve testing efficiency, Test Case Selection (TCS), Test Case Prioritization (TCP), and Test Case Minimization (TCM) strategies have been developed [3]. These techniques help reduce redundant test cases, optimize testing resources, and improve fault detection capability, making regression testing optimization an important research area [4-6].

In this context, TCS aims to select a subset of relevant test cases to re-execute based on software changes. Although often discussed together with TCP and TCM, TCS differs in its goal and scope, focusing on selecting a subset of test cases for a specific regression cycle [4, 5]. Studies primarily addressing TCS were included. Studies involving TCP or TCM were included only when test case selection constituted a substantive component of the proposed approach. Studies focusing exclusively on TCP, TCM, test data generation, or unrelated testing activities were excluded.

Traditional TCS approaches typically rely on deterministic heuristics or single-objective formulations such as coverage or cost. Although easy to implement, these approaches fail to address conflicts between implementation cost and fault detection effectiveness when the search space becomes large and complex [7]. Empirical findings indicate that no single technique is consistently superior across different situations [8-10].

Recent studies have increasingly applied Search-Based Software Testing (SBST) and metaheuristic optimization techniques to address these challenges, particularly in large-scale or industrial environments where multi-objective trade-offs are common [2, 11].

Despite these advances, the literature still lacks a structured synthesis of optimization paradigms, objectives, datasets, and evaluation metrics used in metaheuristic-based TCS research. Therefore, a Systematic Literature Review (SLR) is conducted to classify and synthesize existing studies and to review current research developments [12-14].

II. Research Method

An SLR method was employed to locate, evaluate, and summarize existing empirical literature that analyses TCS by employing metaheuristics for the purposes of regression testing. The research has been conducted in an organized and open

*Corresponding author.

manner, using a documented research protocol that allows for repeatability, accuracy, and consistency of results.

A. Research Questions

The SLR was guided by the following research questions.

RQ1: What types of metaheuristic algorithms are used in test case selection for regression testing?

RQ2: What optimization objectives, evaluation criteria and metrics, and empirical evaluation contexts are used in metaheuristic-based test case selection studies?

RQ3: What are the main empirical findings and current research trends in metaheuristic-based test case selection?

The above research questions help to provide a systematic synthesis of the algorithm's characteristics and how objectives were formulated, experimental contexts, and empirical findings in relation to metaheuristic-based TCS research.

B. Sources of Information

To perform a thorough literature review, multiple key academic databases were consulted to identify and include high-quality studies. The sources of information used in this study include IEEE Xplore, Scopus, ScienceDirect, SpringerLink, and Google Scholar. The selection of these databases was based on their coverage of the software engineering field, publisher reputation, and accessibility to journal articles and conference proceedings reporting empirical evidence.

C. Search Strategy

The search strategy was developed using keywords related to TCS and metaheuristic optimization techniques. The main keywords included “test case selection”, “test suite selection”, and “test suite reduction”, combined with metaheuristic techniques such as “metaheuristic”, “genetic algorithm”, “particle swarm optimization”, “ant colony optimization”, “simulated annealing”, and “NSGA-II”. Boolean operators “AND” and “OR” were used to combine the keywords and control the search scope. The search was limited to English-language journal articles and conference papers reporting empirical studies related to regression testing. Table I presents the search protocol, including the databases, search strings, and publication years covered in this systematic literature review.

TABLE I. SEARCH PROTOCOL

Database	Search String	Year Covered
IEEE Xplore	("test case selection" OR "test suite selection" OR "test suite reduction") AND ("metaheuristic" OR "genetic algorithm" OR "particle swarm optimization" OR "ant colony optimization" OR "simulated annealing" OR "NSGA-II")	2014–2026
Scopus		
ScienceDirect		
SpringerLink		
Google Scholar		

D. Study Selection

The systematic process used for selecting studies in this SLR ensured that only studies relevant to metaheuristic-based TCS were included. Fig. 1 shows the study selection process from the initial identification of records to the final set of primary studies. A total of 323 records were identified through searches of five major databases in the initial stage. The titles, authors, and Digital Object Identifiers (DOIs) of the retrieved records were examined to identify duplicate records and clearly irrelevant studies. A total of 38 duplicate or clearly irrelevant records were removed, leaving 285 records for title and abstract screening.

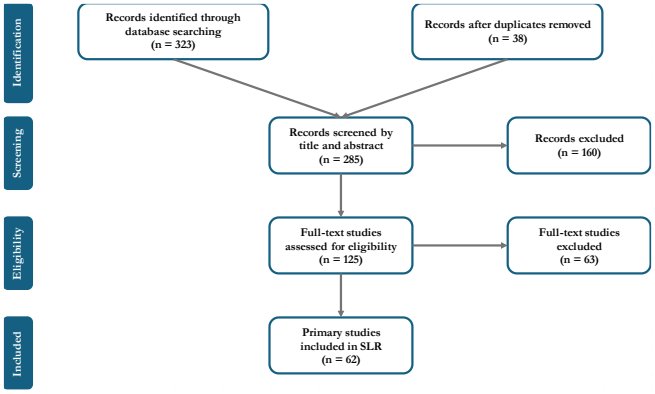

Fig. 1. Overview of the study screening and selection procedure.

In the second stage, the 285 records were screened based on their titles and abstracts to assess their relevance to regression testing and the involvement of TCS components. Studies that focused on TCP and TCM, as well as studies without sufficient empirical evidence, were excluded. A total of 160 records were excluded at this stage, leaving 125 studies for full-text assessment.

In the final stage, the 125 potentially eligible studies underwent full-text assessment based on the clarity of the proposed method, the appropriateness of the evaluation measures, and the quality and completeness of experimental reporting. A total of 63 studies were excluded because they did not satisfy the predefined eligibility and quality criteria, including insufficient methodological description and inadequate empirical evaluation. Consequently, 62 primary studies were selected for data extraction and synthesis to address the research questions.

E. Extraction and Synthesis

To ensure consistency with the study objectives, a data extraction and synthesis process was conducted for each of the 62 primary studies. A standardized template was used to capture data on the metaheuristic algorithm type, TCS strategy, optimization objectives, and empirical evaluation methods [15]. The main extraction items are presented in Table II. The detailed information extracted from the 62 primary studies is provided in Appendix (Table VI).

The predetermined data extraction criteria helped enhance the accuracy and consistency of the extracted data. Any discrepancies identified during the extraction process were discussed and resolved by the reviewers. Data synthesis was conducted in structured and narrative forms to capture the diversity of previous studies. Metaheuristic algorithms, optimization objectives, and experimental designs were identified to support grouping and comparative analysis based on the metaheuristic technique, TCS methodology, and reported performance metrics.

TABLE II. DATA EXTRACTION ITEMS FOR METAHEURISTIC-BASED TEST CASE SELECTION STUDIES

Categories	Data Items	Description
Study Information	Study ID	A unique identifier was allocated to each primary study
	Authors & Year	Authors' names and year of publication
	Publication Source	Venue of publication and conference proceedings
Algorithm	Metaheuristic Algorithm	Type of metaheuristic technique applied within the corresponding optimization problem
	Algorithm Classification	Optimization categories including single-objective and multi-objective
TCS Approach	Selection Strategy	TCS technique or mechanism applied
	Technique Integration	Approaches used independently and those integrated with TCM and TCP
Objectives & Metrics	Optimization Objectives	Types of objective functions considered by the researchers when searching for an optimal solution
	Evaluation Metrics	Types of performance metrics used to evaluate the search results
Empirical Evaluation	Evaluation Dataset	Software systems or benchmark datasets utilized in the study
	Experimental Context	Experimental context includes industrial settings, open-source environments, and simulation platforms
Results	Key Findings	Significant findings reported and contributions to the field
Study Limitations	Limitations	Limitations identified by the authors

F. Quality Assessment of Primary Studies

To assess the quality of the selected primary studies systematically, the full-text studies were evaluated based on three main criteria. These criteria included the clarity of the proposed method, the appropriateness of the evaluation measures, and the quality of experimental reporting. The assessment was also used to determine whether each study provided sufficient methodological and empirical information to support the synthesis of findings. In addition, the completeness of the descriptions of the metaheuristic algorithm, experimental context, datasets, and evaluation metrics was considered. Studies with insufficient empirical evidence or inadequate reporting were excluded during the full-text screening stage.

The review of the 62 primary studies indicates that researchers have proposed numerous methods of measurement to address the problem of TCS. However, different empirical methods and descriptions of TCS mechanisms have resulted in varied criteria for selecting algorithms, making meaningful and direct comparison between studies difficult.

The analysis also shows that many studies use a limited number of evaluation metrics and relatively small test suites. This limitation may affect the generalizability of research findings and the development of hybrid approaches that combine multiple TCS strategies.

These findings indicate the need for clearer specification of TCS approaches to support better characterization and evaluation of TCS methodologies. Improved reporting would help explain how TCS techniques perform in larger-scale environments, particularly in terms of cost, coverage, and fault detection capability. Table III highlights the importance of more standardized reporting and evaluation practices to strengthen the interpretability and comparability of future TCS research.

TABLE III. MAPPING OF QUALITATIVE ASSESSMENT RESULTS TO RESEARCH QUESTIONS

Research Question	Focus of Assessments	Key Qualitative Findings	Identified Implications
RQ1	Metaheuristic techniques for TCS	Algorithms vary widely, and selection mechanisms are often unclear	Need clearer TCS formulations
RQ2	Objectives, metrics, and evaluation contexts	Different objectives, metrics, and datasets are used	Limited comparability across studies
RQ3	Empirical findings and research trends	Many studies report cost reduction and improved effectiveness	Limited generalizability of findings

III. RESULTS

This section reports the findings of a systematic literature review based on the analysis of 62 primary studies examining the use of metaheuristics in TCS. The findings describe the optimization algorithms applied, objective formulations, evaluation metrics, and experimental contexts used to assess the proposed approaches. The synthesis is organized according to the research questions (RQ1-RQ3) to present the findings systematically.

A. Primary Studies

The 62 primary studies identified during the screening are discussed in this section. These studies provide an overview of the metaheuristic techniques used, objective formulations, and evaluation environments applied in TCS research. The distribution of primary studies across digital libraries is presented in Table IV.

TABLE IV. DISTRIBUTION OF PRIMARY STUDIES BY DIGITAL LIBRARY

Digital Library	Journal	Conference	Total
Scopus	17	18	35
IEEE Xplore	2	5	7
SpringerLink	4	2	6
ScienceDirect	2	0	2
Google Scholar	12	0	12
Total of Primary Studies			62

B. Analysis of the Primary Studies

A systematic mapping approach was used to analyze each of the primary studies as they related to the desired research questions RQ1-RQ3. As a result, the study types used, how objectives were formulated, the empirical contexts chosen, and the effectiveness of each technique could be systematically synthesized based on their research findings.

C. RQ1: What Types of Metaheuristic Algorithms Are used in Test Case Selection for Regression Testing?

RQ1 is addressed by presenting empirical evaluations of metaheuristic techniques used for TCS in regression testing. The

analysis focuses on the categories of metaheuristic algorithms applied to the TCS problem and the variations introduced across different studies.

1) Refinement of algorithms: Metaheuristic techniques identified in previous studies can generally be categorized into several groups, including Evolutionary Algorithms (EA), Swarm Intelligence approaches, and other nature-inspired metaheuristics, as summarized in Table V. These categories represent different search strategies adapted to address the complexity of TCS in regression testing.

The EA, particularly Genetic Algorithm (GA) and Non-Dominated Sorting Genetic Algorithm II (NSGA-II), are the most widely applied methods in TCS research for solving multi-objective optimization problems that balance test effectiveness and implementation cost [16-20]. Several studies introduced improvements such as linkage learning-based crossover, seeding strategies, metamorphic testing integration, and operator-level improvements including mutation and crossover strategies to enhance solution stability and performance [19-22].

In addition to evolutionary approaches, swarm intelligence algorithms such as Particle Swarm Optimization (PSO) are also widely used. Early studies focused on Binary Multi-Objective PSO (BMOPSO) and compared its variants [8]. Later work explored hybrid approaches such as BMOPSO combined with Local Search and PSO integrated with Harmony Search to improve the trade-off between coverage and execution cost [23, 24]. Advanced variants such as Quantum-Behaved PSO (QPSO) have also been reported to improve performance over classical approaches [25]. More recent studies further demonstrate the effectiveness of hybrid swarm GA and swarm-based minimization strategies in improving fault detection and execution efficiency [26-31].

Other metaheuristic alternatives explored in previous studies include Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Whale Optimization Algorithm (WOA), Cuckoo Search, Bat Algorithm, and Firefly Algorithm. These techniques are often evaluated comparatively or modified to balance fault detection capability with execution time [9, 32-38]. Variations such as Harmony Search, Firefly Algorithm, and ABC have also been combined with crossover or ranking mechanisms to improve convergence speed and solution quality [28, 39].

Recent studies have proposed advanced metaheuristic algorithms to address scalability and high-dimensional objective spaces, particularly in large-scale regression testing environments [40-42]. Examples include Chaotic Archimedes Optimization Algorithm (CAOA), African Buffalo Optimization (ABO), and Gray Wolf Optimization (GWO), which reflect the expanding exploration of optimization techniques in TCS research [43-45]. These developments indicate that future algorithmic innovation in TCS is likely to focus more on domain-specific adaptations rather than purely theoretical exploration.

2) Uniqueness of the algorithm: The uniqueness of the algorithm proposed in previous studies does not lie in the selection of algorithms alone, but in the way in which the algorithms are adapted to the needs of regression testing. Many studies introduced hybrid strategies, diversity injection mechanisms, or operator improvements to enhance search quality and solution stability. Some studies also proposed optimization-oriented approaches that are not purely metaheuristic, such as Multi-Criteria Decision Making (AHP-TOPSIS), ontology-based selection, belief propagation for microservices, and learning-based techniques such as recurrent neural networks. These approaches highlight the need to adapt test selection techniques to increasingly complex system architectures and testing environments [46-50]. To provide a clearer overview of the metaheuristic techniques identified in this review, Table V summarizes the major algorithm categories, their characteristics, and application contexts.

TABLE V. SUMMARY OF METAHEURISTIC TECHNIQUES USED FOR TEST CASE SELECTION

Author	Type of Algorithm	Key Characteristics	Application Context
[16-20, 22,51-55]	Evolutionary Algorithms (GA, NSGA-II)	Multi-objective, Pareto-based optimization	Benchmark, black-box, industrial systems
[8, 23-25, 27, 28, 30, 38]	Swarm Intelligence Algorithms (PSO, QPSO)	Fast convergence, population-based search	Regression TCS, hybrid approaches
[9,34, 35, 37-39, 43, 44]	Other Nature-Inspired Metaheuristic Algorithms (ACO, ABC, Whale, Bat, Firefly, Cuckoo)	Optimization techniques balancing fault detection and execution cost	Comparative experimental studies and benchmark environments
[43-45, 56, 57]	Emerging Metaheuristic Techniques (CAOA, ABO, GWO)	Adaptive and domain-specific optimization strategies for large-scale testing environments	Recent benchmark studies and advanced regression testing contexts

Fig. 2 illustrates the distribution of metaheuristic algorithm categories across publication years. Evolutionary algorithms, particularly GA and NSGA-II, remain the most dominant approaches for multi-objective TCS. Swarm intelligence algorithms such as PSO and QPSO are also widely used through various variants and hybrid implementations. Other nature-inspired algorithms, including Bat, Firefly, and Cuckoo Search, are mainly applied for comparative evaluation or algorithm modification. The emergence of newer techniques indicates increasing interest in adaptive and domain-specific optimization approaches for regression testing.

D. RQ2: What Optimization Objectives, Evaluation Criteria and Metrics, and Empirical Evaluation Contexts are used in Metaheuristic-Based Test Case Selection Studies?

This section presents a classification of TCS techniques based on the formulation of optimization objectives, evaluation metrics, and empirical evaluation contexts reported in previous studies. The analysis shows that variations in techniques are influenced by the optimization approach used, while the modelling of the TCS problem and the empirical evaluation settings also play an important role in determining the classification structure.

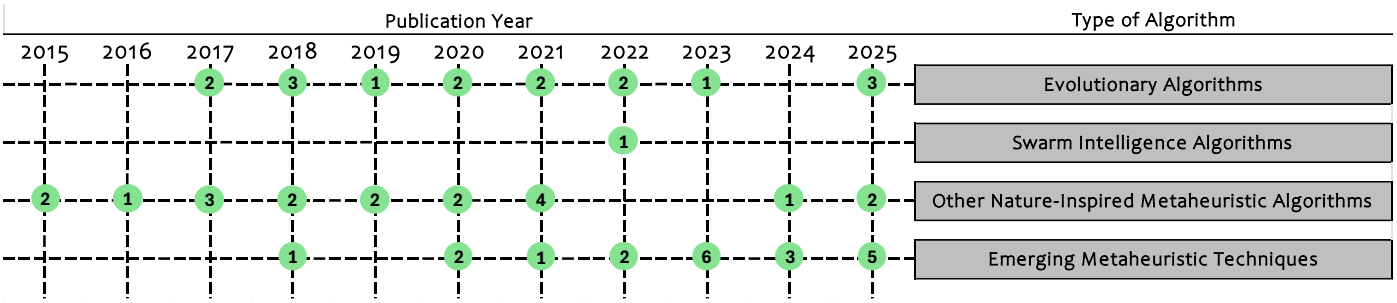

Fig. 2. Distribution of metaheuristic algorithm categories by publication year.

1) Optimization objectives: In general, existing studies can be classified into two broad categories, namely single-objective and multi-objective studies. Single-objective studies typically consolidate multiple criteria into a single objective function and tend to focus on reducing regression testing costs such as execution time and test suite size [9, 30, 51-53].

In contrast, a multi-objective approach allows simultaneous optimization of test effectiveness and implementation cost. This formulation combines difficult-to-measure objectives such as fault detection, test diversity, and coverage with cost factors including test suite size, execution time, and overall cost. The result is a Pareto-optimal solution set that provides trade-off options and supports decision-making in complex regression testing environments [8, 16, 29, 39, 45, 54-56].

Recent studies continue to extend this framework by addressing scalability and high-dimensional objective spaces in large-scale or industrial environments. Research in this domain focuses on optimizing multiple conflicting objectives, including diversity, coverage, fault detection capability, and cost, while maintaining solution quality through Pareto dominance techniques [40-42, 57].

Several studies also integrate additional criteria such as diversity, risk, similarity reduction, and technical debt to support more comprehensive optimization. These criteria enhance the applicability of TCS in specific contexts, particularly for cost-effective optimization strategies [58]. However, variations in objectives and evaluation metrics make it difficult to compare techniques consistently across studies, highlighting the need to align TCS strategies with system-specific requirements and testing constraints [16, 46, 59].

2) Evaluation criteria and performance metrics: In terms of evaluation criteria, the majority of studies evaluate TCS techniques based on two main categories, which are test effectiveness and implementation cost. Test effectiveness is usually evaluated through coverage, for example, statement, path, and interaction coverage, and fault detection metrics such as Fault Detection Rate (FDR) and Fault Detection Capability (FDC). Implementation cost is usually measured through Execution Time (ET), Test Suite size, or Suite Size Reduction, as these metrics directly represent the burden of regression testing in practice [9, 30, 33, 34, 40, 53, 57, 60-62].

The majority of multi-objective studies rely on Pareto-based indicators such as Hypervolume (HV), Generational Distance (GD), Inverted Generational Distance (IGD), epsilon, and Pareto-front characteristics to measure the quality of the resulting solution set. In contrast, single-objective studies tend to rely on performance metrics such as execution time, fault detection rate, Average Percentage of Faults Detected (APFD), and test suite size, as these metrics are generally easier for industry practitioners to interpret [32, 51-53, 55, 63]. These Pareto indicators evaluate solution quality from the perspective of trade-offs and objective space coverage and allow studies to provide alternative solutions for users under different constraints [8, 40-42, 54, 63-65].

Fig. 3 shows the continued dominance of multi-objective formulations in metaheuristic-based TCS studies, reflecting the need to balance test effectiveness and implementation costs.

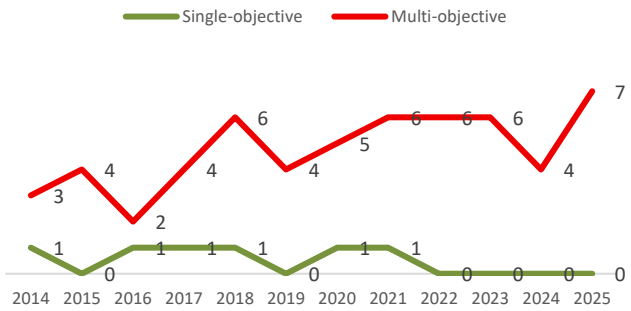

Fig. 3. Multi-objective formulations in metaheuristic-based TCS studies.

Variation in evaluation criteria across studies limits the consistent comparison of empirical results and highlights the need for more standardized evaluation practices in TCS research. Industry studies often prioritize simple metrics such as runtime and fault detection, while benchmark studies rely on Pareto indicators. The introduction of criteria such as diversity, similarity, and technical debt also reflects a shift toward more contextual evaluation metrics, although it increases evaluation heterogeneity across studies [16, 17, 46, 66].

3) Empirical evaluation contexts: This section analyses the empirical evidence used to evaluate the effectiveness of TCS techniques in regression testing. The analysis focuses on the type of empirical evidence, such as systems, datasets, and experimental contexts, used across studies. Overall, the findings show that benchmarks and open-source systems still dominate evaluation due to their ease of replication and comparison. However, a growing body of research evaluates these techniques in industrial, simulation-based, cyber-

physical, and AI-based contexts [16, 18, 40, 55, 57, 61, 63, 66-69].

The majority of studies have used benchmarks and open-source systems, particularly programs from the Software Infrastructure Repository (SIR), because they provide controlled and repeatable environments suitable for comparing techniques across studies. These environments allow researchers to evaluate objectives such as coverage, Fault Detection (FD), and ET in a consistent setting, making SIR the dominant empirical basis for evaluating metaheuristic-based TCS [9, 10, 23, 24, 34, 50, 58, 64, 70, 71]. However, this reliance on benchmark environments may limit external validity.

To address this limitation, several studies extend evaluation to larger-scale open-source systems or multiple domains, including enterprise systems and black-box or simulation contexts. These evaluations aim to demonstrate the scalability and feasibility of proposed techniques under more realistic testing conditions [16, 17, 40, 56, 57, 63, 72].

Several studies also report empirical evidence from specific industrial domains such as defense, Programmable Logic Controllers (PLCs), industrial applications, and automotive systems. These evaluations emphasize practical considerations including runtime efficiency, stability of optimization results, and the ability to operate under test time and budget constraints [61, 68]. Recent studies also involve AI-based systems such as deep learning, reinforcement learning, and microservices architectures [48, 69]. These contexts require adaptations in test construction and selection strategies, for example through metamorphic relations or dependency modelling. This trend indicates that TCS research is expanding beyond traditional benchmark environments [18, 26, 47, 48, 55, 60, 67-69, 73].

E. RQ3: What Are the Main Empirical Findings and Current Research Trends in Metaheuristic-Based Test Case Selection?

This section compares TCS techniques based on empirical findings, focusing on regression testing cost reduction and fault detection effectiveness. The analysis shows that technique performance depends on system context, objective formulation, and evaluation criteria, indicating that no single technique is consistently superior across different scenarios.

1) Cost reduction: Most studies report significant reductions in execution time or test suite size compared to conventional approaches such as random selection or retest-all. Metaheuristic-based techniques, particularly those focused on cost minimization, reduce the burden of regression testing without executing the entire test suite [9, 27, 29, 30, 36, 51-53, 58].

Single-objective approaches often achieve greater cost reductions because the optimization focuses on a single criterion. However, higher cost reduction may reduce fault detection effectiveness when test effectiveness is not considered [9, 30, 52, 53]. Multi-objective approaches provide more balanced solutions by allowing users to select Pareto solutions based on time constraints or testing budgets [41, 54, 55].

Techniques such as hybridization, local search integration, and adaptive mutation operators further improve cost efficiency and are suitable for environments such as Continuous Integration (CI) and industrial regression testing [21, 23, 40-42, 57, 60, 72]. Overall, these findings highlight a trade-off between cost efficiency and testing effectiveness, emphasizing the need for balanced optimization strategies.

2) Fault detection effectiveness: Most studies show that metaheuristic-based TCS techniques improve fault detection compared to conventional approaches. Even when only a subset of test cases is executed, these techniques can achieve similar or higher fault detection probability, as demonstrated through metrics such as FDR, FDC, and APFD [16, 25, 27-29, 32, 52, 53, 58].

Multi-objective approaches that model fault detection or coverage as explicit objectives generally provide more stable defect detection performance. Techniques incorporating diversity, similarity reduction, or metamorphic relations help select more representative and less redundant test sets, increasing the likelihood of detecting diverse defects, particularly in complex systems such as deep learning and cyber-physical systems [16, 18, 59, 74].

However, empirical findings show that no single technique consistently outperforms others in fault detection. Performance depends on system characteristics, fault types, and metric definitions. Several industrial studies also emphasize that repeatability and practical feasibility are often more important than maximum performance in controlled environments [40, 55, 57, 60, 72]. These findings indicate that effectiveness in metaheuristic-based TCS is inherently context-dependent rather than universally achieved through a single optimization strategy.

3) Synthesis of trade-offs in metaheuristic-based TCS: The findings indicate an inherent trade-off between test case execution cost and fault detection effectiveness in regression testing. This trade-off has led to the development of hybrid techniques to achieve a better balance between cost and effectiveness. Fig. 4 illustrates this conceptual trade-off in metaheuristic-based TCS.

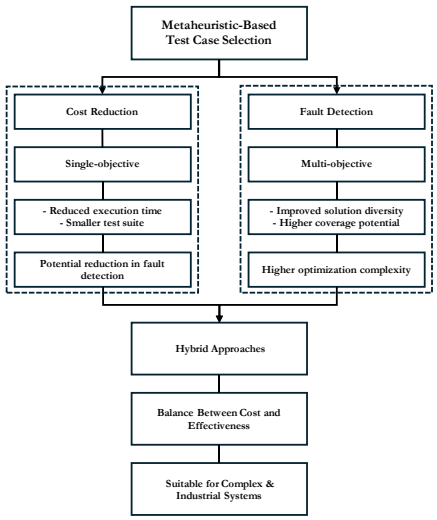

Fig. 4. Conceptual synthesis of the cost-effectiveness trade-off in metaheuristic-based test case selection.

IV. Discussion

Based on the findings of RQ1 in Section III-C, the dominance of evolutionary algorithms and swarm intelligence is driven more by flexibility in modelling regression testing trade-offs than by absolute algorithmic superiority [15, 16, 51]. The shift towards hybrid techniques and operator improvement confirms that the adaptation of algorithms to the problem context is more significant than the mere selection of basic algorithms [22, 24, 25].

For RQ2 in Section III-D, there is a clear difference between single-objective and multi-objective approaches. Multi-objective approaches evaluated using Pareto indicators provide more flexible decision support but increase the complexity of the analysis and make comparisons between studies difficult [8, 51, 55]. Time-constrained environments such as industrial settings are typically more suited to cost-oriented approaches [9, 53].

For RQ3 in Section III-E, there is a predominance of benchmarking and open-source systems from the SIR [9, 24]. Although benchmarking establishes a basis for replication, the considerable reliance on benchmarks for assessing and comparing evidence may constrain the external validity of the associated findings. Similarly, industrial environments, cyber-physical systems, and AI-enabled environments are increasingly being assessed within an industrial context. Each of these three environments has substantial variances and, consequently, these variances greatly affect how metrics are interpreted and how successful the metrics are at measuring what they are designed to measure [17, 51, 58].

Taken together, these findings suggest that the suitability of a TCS approach should be determined by the primary testing objective and the constraints of the target environment rather than by algorithmic popularity alone. For cost-oriented TCS, single-objective or cost-focused approaches are more suitable when reducing execution time or test suite size is the primary concern, particularly in industrial and time-constrained environments. Coverage-oriented TCS is better suited to coverage-based evolutionary approaches when maximizing structural or interaction coverage is the dominant objective. For fault-detection-oriented TCS, detection-based or multi-objective approaches may be preferable because they can prioritize fault detection while considering competing objectives. When both cost and test effectiveness are important, multi-objective algorithms are more appropriate because they explicitly model trade-offs and provide alternative solutions under different testing constraints. For many-objective TCS problems, many-objective evolutionary or hybrid approaches may be more suitable because they are designed to manage a larger number of competing objectives. In industrial or time-constrained settings, cost-aware and hybrid approaches may offer greater practical suitability by balancing solution quality with computational and execution overhead.

Furthermore, the findings also highlight the perceived trade-off between detecting defects and reducing costs. Cost-based strategies tend to achieve large reductions in costs but may compromise their ability to detect faults [9, 31]. Multi-objective and hybrid techniques may provide a more balanced trade-off when multiple objectives must be considered simultaneously [51, 55]. The findings indicate that no single approach is consistently superior across all scenarios, with a discernible trend toward more adaptive and context-sensitive techniques [15, 22]. This suggests that effective TCS strategies depend on aligning the optimization method with the testing objective, system characteristics, and operational constraints.

V. Threats to Validity

A. Publication Bias

Publication bias may occur because studies with significant, positive, or novel findings are more likely to be published than studies reporting non-significant or negative results. As a result, the studies included in this review may not fully reflect the overall body of research on the topic. To reduce this risk, the review considered studies from multiple sources and applied predefined inclusion and exclusion criteria.

B. Database Selection Bias

The selection of bibliographic databases may have influenced the studies identified for this review. Different databases have varying levels of coverage, indexing practices, and disciplinary focuses. Therefore, relevant studies indexed outside the selected databases may have been overlooked. Although the databases were selected based on their relevance to the research topic, this limitation may still affect the completeness of the literature identified.

C. Search-String Limitations

The search strategy depends on the keywords, synonyms, and Boolean operators included in the search string. As a result, relevant studies may have been missed if they used terminology that differed from the terms adopted in this review. Although the search string was developed to capture different variations of the key research concepts, it cannot guarantee that all potentially relevant publications were identified.

D. Study Selection and Exclusion Bias

The process of screening and selecting studies may involve researcher judgment, particularly when determining whether a study meets the inclusion and exclusion criteria. To reduce this risk, predefined eligibility criteria were applied consistently throughout the screening process. However, the possibility of inadvertent exclusion or subjective interpretation during study selection cannot be eliminated.

E. Classification Subjectivity

The classification and categorization of the selected studies may involve some degree of subjectivity. Studies addressing similar concepts may use different terminology, theoretical perspectives, or methodological approaches, which can make their classification dependent on the researchers' interpretation. Clear classification criteria were applied to improve consistency, although some degree of judgment remains unavoidable.

F. External Validity and Generalizability

The generalizability of the findings may be limited by the characteristics of the studies included in the review, including their research contexts, geographical settings, populations, methodologies, and publication periods. Therefore, the findings

should be interpreted within the scope of the reviewed literature and may not be directly applicable to all contexts or populations. Future research involving broader contexts and additional empirical evidence may help strengthen the external validity and generalizability of the findings.

VI. Conclusion and Future Work

This study presents an SLR of metaheuristic-based techniques for TCS in the context of regression testing. The review examines the types of techniques employed, as well as the objectives and metrics used in their formulation and evaluation. It also investigates the empirical context in which these techniques are evaluated and compared. TCS has generally been formulated as an optimization problem that seeks to balance test effectiveness with the cost of regression testing, as presented in Section III and discussed in Section IV. However, the empirical evidence indicates that no single technique is universally superior. Instead, performance is strongly influenced by the characteristics of the system under test, the objectives being optimized, and the evaluation criteria adopted.

In terms of empirical evidence, evaluation remains largely dominated by benchmark datasets and open-source systems, despite a growing body of research involving industrial contexts, cyber-physical systems, and AI-based systems. Such contextual differences affect the choice of evaluation metrics and the interpretation of technique effectiveness, thereby limiting the comparability and generalizability of findings across studies. Consequently, benchmark performance should not be considered a direct indicator of practical effectiveness, particularly when TCS techniques are applied to systems with different workloads, constraints, and operational requirements.

These findings have implications for both researchers and practitioners. For researchers, future studies should establish a stronger alignment between optimization objectives and evaluation metrics, while adopting more standardized metrics and reporting practices to improve comparability and reproducibility. For practitioners, the selection of a TCS technique should consider not only its reported benchmark performance but also its suitability for the target system, testing objectives, computational constraints, and operational environment. In particular, empirical results obtained from benchmark or open-source systems should be validated before being generalized to industrial settings.

Overall, this study indicates that TCS research is gradually moving beyond the development of new algorithms towards the adaptation and contextualization of existing metaheuristic techniques to address the requirements of regression testing and real-world environments. This shift emphasizes the need for context-aware optimization, standardized evaluation, and stronger industrial validation. Addressing these aspects can help bridge the gap between benchmark-based research and practical deployment, thereby supporting the development and adoption of metaheuristic-based TCS techniques that are more robust, comparable, and applicable across diverse software testing domains.

As directions for future research, industry-scale empirical evaluation needs to be strengthened to improve the external validity of findings and to assess factors such as scalability, computational cost, and effectiveness under realistic testing conditions. In parallel, the development of domain-aware and adaptive TCS techniques is increasingly important for modern systems, particularly AI-based and cyber-physical systems, where conventional assumptions and evaluation criteria may not adequately capture system-specific testing requirements. Such approaches should account for domain characteristics when defining optimization objectives, selecting evaluation metrics, and interpreting TCS effectiveness.

Acknowledgment

This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through GPPS Vot (J168); and the communication of this research was made possible through monetary assistance by Universiti Tun Hussein Onn Malaysia and the UTHM Publisher's Office via Publication Fund E15216.

References

[1] L. B. R. dos Santos, E. F. de Souza, A. T. Endo, C. Trubiani, R. Pinciroli, and N. L. Vijaykumar, "Performance regression testing initiatives: a systematic mapping," Inf. Softw. Technol., vol. 179, no. November 2024, p. 107641, 2025, doi: 10.1016/j.infsof.2024.107641.

[2] A. Bajaj and O. P. Sangwan, "A Survey on Regression Testing Using Nature-Inspired Approaches," 2018 4th International Conference on Computing Communication and Automation, ICCCA 2018, pp. 1–5, 2018, doi: 10.1109/CCAA.2018.8777692.

[3] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," Software Testing, Verification and Reliability, vol. 22, no. 2, pp. 67–120, 2012, doi: 10.1002/stvr.430.

[4] S. Parida, D. Rath, and D. B. Mishra, "A Review on Test Case Selection, Prioritization and Minimization in Regression Testing," Artificial Intelligence-Enhanced Software and Systems Engineering, vol. 1, pp. 156–163, 2022.

[5] E. N. Narciso, M. E. Delamaro, and F. De Lourdes Dos Santos Nunes, "Test Case Selection: A Systematic Literature Review," International Journal of Software Engineering and Knowledge Engineering, vol. 24, no. 4, pp. 653–676, 2014, doi: 10.1142/S0218194014500259.

[6] R. Kazmi, D. N. A. Jawawi, R. Mohamad, and I. Ghani, "Effective Regression Test Case Selection: A Systematic Literature Review," ACM Comput. Surv., vol. 50, no. 2, 2018, doi: 10.1145/3057269.

[7] M. Rehan et al., "A Systematic Analysis of Regression Test Case Selection: A Multi-Criteria-Based Approach," Security and Communication Networks, vol. 2021, 2021, doi: 10.1155/2021/5834807.

[8] L. S. De Souza, R. B. C. Prudencio, and F. D. A. Barros, "A Comparison Study of Binary Multi-Objective Particle Swarm Optimization Approaches for Test Case Selection," Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, pp. 2164–2171, 2014, doi: 10.1109/CEC.2014.6900522.

[9] A. Kaur and A. P. Agrawal, "A Comparative Study of Bat and Cuckoo Search Algorithm for Regression Test Case Selection," Proceedings of the 7th International Conference Confluence 2017 on Cloud Computing, Data Science and Engineering, pp. 164–170, 2017, doi: 10.1109/CONFLUENCE.2017.7943143.

[10] A. Choudhary, A. P. Agrawal, and A. Kaur, "An Effective Approach for Regression Test Case Selection using Pareto based Multi-Objective Harmony Search," Proceedings - International Conference on Software Engineering, pp. 13–20, 2018, doi: 10.1145/3194718.3194722.

[11] N. Prabhakar, A. Singhal, A. Bansal, and V. Bhatia, "A Literature Survey of Applications of Meta-heuristic Techniques in Software Testing," Advances in Intelligent Systems and Computing, vol. 731, pp. 497–505, 2019.

[12] Z. Xiao and L. Xiao, "A Systematic Literature Review on Test Case Prioritization and Regression Test Selection," Proceedings - 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications, SERA 2023, no. 2022, pp. 235–242, 2023, doi: 10.1109/SERA57763.2023.10197719.

[13] H. Singh, L. Singh, and S. Tiwari, "A Systematic Literature Review on Test Case Prioritization Techniques," International Journal of Software Innovation, vol. 10, no. 1, pp. 1–36, 2022, doi: 10.4018/IJSI.312263.

[14] R. Pan, M. Bagherzadeh, T. A. Ghaleb, and L. Briand, "Test Case Selection and Prioritization Using Machine Learning: A Systematic Literature Review," Empir. Softw. Eng., vol. 27, no. 2, pp. 1–34, 2022, doi: 10.1007/s10664-021-10066-6.

[15] C. Wohlin, E. Mendes, K. R. Felizardo, and M. Kalinowski, "Guidelines for the search strategy to update systematic literature reviews in software engineering," Inf. Softw. Technol., vol. 127, no. January, p. 106366, 2020, doi: 10.1016/j.infsof.2020.106366.

[16] D. Mondal, H. Hemmati, and S. Durocher, "Exploring Test Suite Diversification and Code Coverage in Multi-Objective Test Case Selection," 2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings, pp. 1–10, 2015, doi: 10.1109/ICST.2015.7102588.

[17] A. Arrieta, U. Markiegi, S. Wang, G. Sagardui, A. Arruabarrena, and L. Etxeberria, "Multi-objective black-box test case selection for cost-effectively testing simulation models," GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference, no. July 2018, pp. 1411–1418, 2018, doi: 10.1145/3205455.3205490.

[18] A. Arrieta, "Multi-Objective Metamorphic Follow-up Test Case Selection for Deep Learning Systems," GECCO 2022 - Proceedings of the 2022 Genetic and Evolutionary Computation Conference, no. January, pp. 1327–1335, 2022, doi: 10.1145/3512290.3528697.

[19] D. Pradhan, S. Wang, S. Ali, T. Yue, and M. Liaaen, "CBGA-ES: A Cluster-Based Genetic Algorithm with Elitist Selection for Supporting Multi-Objective Test Optimization," Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, pp. 367–378, 2017, doi: 10.1109/ICST.2017.40.

[20] M. Huang, S. Guo, X. Liang, and X. Jiao, "Research on Regression Test Case Selection Based on Improved Genetic Algorithm," Proceedings of 2013 3rd International Conference on Computer Science and Network Technology, ICCSNT 2013, pp. 256–259, 2014, doi: 10.1109/ICCSNT.2013.6967108.

[21] M. Ugarte, P. Valle, M. Illarramendi, and A. Arrieta, "Enhancing multi objective test case selection through the mutation operator," Automated Software Engineering, vol. 32, no. 1, pp. 1–24, 2025, doi: 10.1007/s10515-025-00489-6.

[22] A. Panichella, R. Oliveto, M. Di Penta, and A. De Lucia, "Improving Multi-Objective Test Case Selection by Injecting Diversity in Genetic Algorithms," IEEE Transactions on Software Engineering, vol. 41, no. 4, pp. 358–383, 2015, doi: 10.1109/TSE.2014.2364175.

[23] L. S. De Souza, R. B. C. Prudencio, and F. A. De Barros, "An Hybrid Binary Multi-Objective Particle Swarm Optimization with Local Search for Test Case Selection," Proceedings - 2014 Brazilian Conference on Intelligent Systems, BRACIS 2014, pp. 414–419, 2014, doi: 10.1109/BRACIS.2014.80.

[24] L. S. de Souza, R. B. Cavalcante Prudencio, and F. A. de Barros, "A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection," Journal of the Brazilian Computer Society, vol. 21, no. 1, pp. 1–20, 2015, doi: 10.1186/s13173-015-0038-8.

[25] A. Bajaj, A. Abraham, S. Ratnoo, and L. A. Gabralla, "Test Case Prioritization, Selection, and Reduction Using Improved Quantum-Behaved Particle Swarm Optimization," Sensors, pp. 1–18, 2022.

[26] Palak, P. Gulia, and N. S. Gill, "An Enhanced Artificial Bee Colony: Naïve Bayes Technique for Optimizing Software Testing," International Journal of Advanced Computer Science and Applications, vol. 12, no. 2, pp. 220–225, 2021, doi: 10.14569/IJACSA.2021.0120228.

[27] Palak and P. Gulia, "Hybrid swarm and GA based approach for software test case selection," International Journal of Electrical and Computer Engineering, vol. 9, no. 6, pp. 4898–4903, 2019, doi: 10.11591/ijece.v9i6.pp4898-4903.

[28] Palak, P. Gulia, and N. S. Gill, "Optimized Test Case Selection using Scout-less Hybrid Artificial Bee Colony Approach and Crossover Operator," International Journal of Engineering Trends and Technology, vol. 69, no. 3, pp. 39–45, 2021, doi: 10.14445/22315381/IJETT-V69I3P208.

[29] M. Bharathi, "Hybrid Particle Swarm and Ranked Firefly Metaheuristic Optimization-Based Software Test Case Minimization," International Journal of Applied Metaheuristic Computing, vol. 13, no. 1, pp. 1–20, 2021, doi: 10.4018/ijamc.290534.

[30] R. Nagar, A. Kumar, S. Kumar, and A. S. Baghel, "Implementing Test Case Selection and Reduction Techniques using Meta-Heuristics," Proceedings of the 5th International Conference on Confluence 2014: The Next Generation Information Technology Summit, pp. 837–842, 2014, doi: 10.1109/CONFLUENCE.2014.6949377.

[31] S. Singhal, B. Suri, and S. Misra, "An Empirical Study of Regression Test Suite Reduction Using MHBG_TCS Tool," Proceedings of the IEEE International Conference on Computing, Networking and Informatics, ICCNI 2017, vol. 2017-Janua, pp. 1–5, 2017, doi: 10.1109/ICCNI.2017.8123805.

[32] D. Silva, R. Rabelo, M. Campanha, P. S. Neto, P. A. Oliveira, and R. Britto, "A Hybrid Approach for Test Case Prioritization and Selection," 2016 IEEE Congress on Evolutionary Computation, CEC 2016, pp. 4508–4515, 2016, doi: 10.1109/CEC.2016.7744363.

[33] P. Palak and P. Gulia, "Ant Colony Optimization Based Test Case Selection for Component Based Software," International Journal of Engineering and Technology (UAE), vol. 7, no. 4, pp. 2743–2745, 2018, doi: 10.14419/ijet.v7i4.17565.

[34] A. P. Agrawal, A. Choudhary, and A. Kaur, "An Effective Regression Test Case Selection Using Hybrid Whale Optimization Algorithm," International Journal of Distributed Systems and Technologies, vol. 11, no. 1, pp. 53–67, 2020, doi: 10.4018/IJDST.2020010105.

[35] P. Harris, "Modified Firefly algorithm for Optimal Test Case Selection," Proceedings - 7th International Conference on Computing Methodologies and Communication, ICCMC 2023, pp. 587–591, 2023, doi: 10.1109/ICCMC56507.2023.10083726.

[36] K. Senthil Kumar and A. Muthukumaravel, "Optimal Test Suite Selection using Improved Cuckoo Search Algorithm Based on Extensive Testing Constraints," International Journal of Applied Engineering Research, vol. 12, no. 9, pp. 1920–1928, 2017.

[37] R. Nagar, A. Kumar, G. P. Singh, and S. Kumar, "Test Case Selection and Prioritization using Cuckoos Search Algorithm," 2015 1st International Conference on Futuristic Trends in Computational Analysis and Knowledge Management, ABLAZE 2015, pp. 283–288, 2015, doi: 10.1109/ABLAZE.2015.7155012.

[38] M. H. Alkawaz, A. Silvarajoo, O. F. Mohammad, and L. Raya, "A System for Optimizing Software Regression Test Cases using Modified Ant Colony Optimization Algorithm," 2022 IEEE Symposium on Industrial Electronics and Applications, ISIEA 2022, pp. 1–5, 2022, doi: 10.1109/ISIEA54517.2022.9873649.

[39] K. Han, Y. Song, and Y. Zhang, "Regression Test Case Selection Based on multi-objective Optimization and Improved Harmonic Search Algorithms," ICICN 2023 - 2023 IEEE 11th International Conference on Information, Communication and Networks, pp. 830–834, 2023, doi: 10.1109/ICICN59530.2023.10393597.

[40] F. Dobslaw, R. Wan, and Y. Hao, "Generic and industrial scale many-criteria regression test selection," Journal of Systems and Software, vol. 205, p. 111802, 2023, doi: 10.1016/j.jss.2023.111802.

[41] U. K. Jaiswal and A. Prajapati, "Many Objective Feedback Evolutionary Algorithm for Optimizing the Software Test Suite," SN Comput. Sci., vol. 6, no. 1, 2025, doi: 10.1007/s42979-024-03580-z.

[42] A. Trovato, M. De Stefano, F. Pecorelli, D. Di Nucci, and A. De Lucia, "Reformulating regression test suite optimization using quantum annealing-an empirical study," International Journal on Software Tools for Technology Transfer, vol. 26, no. 6, pp. 767–780, 2024, doi: 10.1007/s10009-024-00775-w.

[43] A. S. Verma, A. Choudhary, and S. Tiwari, "A novel chaotic archimedes optimization algorithm and its application for efficient selection of regression test cases," International Journal of Information Technology (Singapore), vol. 15, no. 2, pp. 1055–1068, 2023, doi: 10.1007/s41870-022-01031-7.

[44] S. Singhal, N. Jatana, A. F. Subahi, C. Gupta, O. I. Khalaf, and Y. Alotaibi, "Fault Coverage-Based Test Case Prioritization and Selection Using African Buffalo Optimization," Computers, Materials and

Continua, vol. 74, no. 3, pp. 6755–6774, 2023, doi: 10.32604/cmc.2023.032308.

[45] S. Kumari, S. Jindal, and A. Sharma, "Test case optimization using grey wolf algorithm," Software Quality Journal, vol. 33, no. 2, 2025, doi: 10.1007/s11219-025-09717-4.

[46] A. Zarrad, R. Bahsoon, and P. Manimaran, "Optimizing regression testing with AHP TOPSIS metric system for effective technical debt evaluation," Automated Software Engineering, vol. 31, no. 2, pp. 1–28, 2024, doi: 10.1007/s10515-024-00458-5.

[47] Z. Sakhrawi and T. Labidi, "Test case selection and prioritization approach for automated regression testing using ontology and COSMIC measurement," Automated Software Engineering, vol. 31, no. 2, pp. 1–32, 2024, doi: 10.1007/s10515-024-00447-8.

[48] L. zhe Chen, J. Wu, H. yan Yang, and K. Zhang, "A microservice regression testing selection approach based on belief propagation," Journal of Cloud Computing, vol. 12, no. 1, 2023, doi: 10.1186/s13677-023-00398-7.

[49] X. Wu, J. Shen, W. Zheng, L. Lin, Y. Sui, and A. O. A. Semasaba, "RNNtcs: A test case selection method for Recurrent Neural Networks," Knowl. Based. Syst., vol. 279, p. 110955, 2023, doi: 10.1016/j.knosys.2023.110955.

[50] B. Alkhazi, C. Abid, M. Kessentini, D. Leroy, and M. Wimmer, "Multi criteria test cases selection for model transformations," Automated Software Engineering, vol. 27, no. 1–2, pp. 91–118, 2020, doi: 10.1007/s10515-020-00271-w.

[51] I. Ghani, W. M. N. Wan Kadir, A. F. Arbain, and I. Ghani, "A Detection-Based Multi-Objective Test Case Selection Algorithm to Improve Time and Efficiency in Regression Testing," IEEE Access, vol. 12, no. July, pp. 114974–114994, 2024, doi: 10.1109/ACCESS.2024.3435678.

[52] J. P. Rajasingh, P. S. Kumar, and S. Srinivasan, "Efficient Fault Detection by Test Case Prioritization via Test Case Selection," Journal of Electronic Testing: Theory and Applications (JETTA), vol. 39, no. 5–6, pp. 659–677, 2023, doi: 10.1007/s10836-023-06086-3.

[53] I. Ghani, W. M. N. Wan-Kadir, A. F. Arbain, and N. Ibrahim, "Improved Test Case Selection Algorithm to Reduce Time in Regression Testing," Computers, Materials and Continua, vol. 72, no. 1, pp. 635–650, 2022, doi: 10.32604/cmc.2022.025027.

[54] T. Saber, F. Delavernhe, M. Papadakis, M. Oneill, and A. Ventresque, "A Hybrid Algorithm for Multi-objective Test Case Selection," 2018 IEEE Congress on Evolutionary Computation, CEC 2018 - Proceedings, pp. 1–8, 2018, doi: 10.1109/CEC.2018.8477875.

[55] V. Garousi, R. Ozkan, and A. Betin-Can, "Multi-objective regression test selection in practice: An empirical study in the defense software industry," Inf. Softw. Technol., vol. 103, no. February, pp. 40–54, 2018, doi: 10.1016/j.infsof.2018.06.007.

[56] N. Gokilavani and B. Bharathi, "Multi-Objective based test case selection and prioritization for distributed cloud environment," Microprocess. Microsyst., vol. 82, no. January, p. 103964, 2021, doi: 10.1016/j.micpro.2021.103964.

[57] R. Lachmann, S. Schulze, M. Felderer, C. Seidl, M. Nieke, and I. Schaefer, "Multi-Objective Black-Box Test Case Selection for System Testing," GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference, no. January, pp. 1311–1318, 2017, doi: 10.1145/3071178.3071189.

[58] L. Raamesh, S. Radhika, and S. Jothi, "A cost-effective test case selection and prioritization using hybrid battle royale-based remora optimization," Neural Comput. Appl., vol. 34, no. 24, pp. 22435–22447, 2022, doi: 10.1007/s00521-022-07627-1.

[59] A. S. Habib, S. U. R. Khan, S. Hussain, N. Ibrahim, H. un Nisa, and A. Yousafzai, "A similarity-based multi-objective test optimization technique using search algorithm," Systems and Soft Computing, vol. 6, no. May 2024, doi: 10.1016/j.sasc.2024.200164.

[60] M. U. Querejeta, E. Jee, L. Liu, P. Valle, A. Arrieta, and M. I. Rezabal, "Search-based Test Case Selection for PLC Systems using Functional Block Diagram Programs," Proceedings - International Symposium on Software Reliability Engineering, ISSRE, no. 2, pp. 228–239, 2023, doi: 10.1109/ISSRE59848.2023.00040.

[61] W. D. F. Mendonça, W. K. G. Assuncao, and S. R. Vergilio, "Feature-oriented test case selection and prioritization during the evolution of highly-configurable systems," Journal of Systems and Software, vol. 217, no. January, p. 112157, 2024, doi: 10.1016/j.jss.2024.112157.

[62] Z. Mafi and S. H. Mirian-Hosseinabadi, "Regression test selection in test driven development," Automated Software Engineering, vol. 31, no. 1, pp. 1–27, 2024, doi: 10.1007/s10515-023-00405-w.

[63] A. Arrieta, S. Wang, U. Markiegi, A. Arruabarrena, L. Etxeberria, and G. Sagardui, "Pareto efficient multi-objective black-box test case selection for simulation-based testing," Inf. Softw. Technol., vol. 114, no. November 2018, pp. 137–154, 2019, doi: 10.1016/j.infsof.2019.06.009.

[64] M. Olsthoorn and A. Panichella, "Multi-objective Test Case Selection Through Linkage Learning-based Crossover," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 12914 LNCS, pp. 87–102, 2021, doi: 10.1007/978-3-030-88106-1_7.

[65] A. Arrieta, J. A. Agirre, and G. Sagardui, "Seeding Strategies for Multi-Objective Test Case Selection: An Application on Simulation-based Testing," GECCO 2020 - Proceedings of the 2020 Genetic and Evolutionary Computation Conference, no. January, pp. 1222–1231, 2020, doi: 10.1145/3377930.3389810.

[66] A. Arrieta, S. Wang, G. Sagardui, and L. Etxeberria, "Search-Based Test Case Selection of Cyber-Physical System Product Lines for Simulation-Based Validation," ACM International Conference Proceeding Series, vol. 16-23-Sept, no. January, pp. 297–306, 2016, doi: 10.1145/2934466.2946046.

[67] N. Medhat, S. M. Moussa, N. L. Badr, and M. F. Tolba, "A Framework for Continuous Regression and Integration Testing in IoT Systems Based on Deep Learning and Search-Based Techniques," IEEE Access, vol. 8, pp. 215716–215726, 2020, doi: 10.1109/ACCESS.2020.3039931.

[68] P. K. Bolisetty, K. R. Akhila, A. Krishna, and J. Rajasekaran, "Object-Oriented and Data Flow based Software Defect Prediction using Hybrid Salp Swarm Algorithm and Genetic Algorithm," 2025 International Conference on Intelligent Computing and Knowledge Extraction,ICICKE 2025, pp. 1–6, 2025, doi: 10.1109/ICICKE65317.2025.11136305.

[69] J. E. Betten, Q. Mazouni, D. Gross, P. Lind, and H. Spieker, "Reusable Test Suites for Reinforcement Learning," Lecture Notes in Computer Science, vol. 16107 LNCS, pp. 125–141, 2026, doi: 10.1007/978-3-032-05188-2_9.

[70] S. Noureen and S. Asghar, "Application of Search Algorithms for Model Based Regression Testing," Research Journal of Applied Sciences, Engineering and Technology, vol. 7, no. 14, pp. 2981–2986, 2014, doi: 10.19026/rjaset.7.630.

[71] M. Benito-Parejo and M. G. Merayo, "Using Genetic Algorithms To Select Test Cases for Finite State Machines with Timeouts," 2021 IEEE Congress on Evolutionary Computation, CEC 2021 - Proceedings, pp. 2403–2410, 2021, doi: 10.1109/CEC45853.2021.9504764.

[72] C. Magalhaes, J. Andrade, L. Perrusi, A. Mota, F. Barros, and E. Maia, "HSP: A hybrid selection and prioritisation of regression test cases based on information retrieval and code coverage applied on an industrial case study," Journal of Systems and Software, vol. 159, 2020, doi: 10.1016/j.jss.2019.110430.

[73] J. Ayerdi, A. Arrieta, E. B. Pobee, and M. Arratibel, "Multi-Objective Metamorphic Test Case Selection: An Industrial Case Study," Proceedings-International Symposium on Software Reliability Engineering, ISSRE, vol. 2022-Octob, pp. 541–552, 2022, doi: 10.1109/ISSRE55969.2022.00058.

[74] M. A. Siddique, W. M. N. Wan-Kadir, J. Ahmad, and N. Ibrahim, "Hybrid Framework to Exclude Similar and Faulty Test Cases in Regression Testing," Baghdad Science Journal, vol. 21, no. 2, pp. 802–811, 2024, doi: 10.21123/bsj.2024.9710.

APPENDIX

TABLE VI. THE DETAILED INFORMATION EXTRACTED FROM THE 62 PRIMARY STUDIES

ID	Authors	Years	Algorithms	Objectives	Datasets	Main Evaluation Metrics
P1	[16]	2015	NSGA-II	Multi-objective	Apache Ant, Apache Derby, JBoss Application Server, NanoXML, Math	Coverage, Diversity, ET, FDR
P2	[17]	2018	NSGA-II	Multi-objective	NSGA-II	HV, ET
P3	[18]	2022	NSGA-II	Multi-objective	Deep Learning Models with MR Test Sets	HV, Pareto-frontier, FD
P4	[19]	2017	CBGA-ES	Multi-objective	Real-World Software Test Suites	Fitness, HV
P5	[20]	2013	Improved GA	Single objective / multi-criteria	Example Software Systems	CC, TCC
P6	[21]	2025	Evolutionary Algorithm (Mutation Enhancement)	Multi-objective	Sample Experimental Datasets	HV, FDR, FDC, Pareto-frontier
P7	[22]	2015	DIV-GA	Multi-objective	SIR	ET
P8	[8]	2014	BMOPSO	Multi-objective	Case Study for Integration Suites and Regression Suites	HV, GD, IGD, C
P9	[23]	2014	Hybrid BMOPSO + Local Search	Multi-objective	SIR	HV, GD, IGD, C
P10	[24]	2015	Hybrid PSO + Harmony Search	Multi-objective	SIR	HV, GD, IGD, C
P11	[25]	2022	Improved QPSO	Multi-objective / multi-criteria	Software Benchmark Programs	APFD, APSC, TSP, FDLP, CRP
P12	[26]	2021	Enhanced ABC-NB Optimization	Optimization / learning-assisted	Software Test Programs - 3	ET, FD
P13	[27]	2019	Hybrid Swarm Intelligence + GA	Single objective / multi-objective	Synthetic Software Testing Data	FD, ET
P14	[28]	2021	Hybrid ABC	Single objective / multi-objective	Open-Source Software Programs	AC, ET, Percentage of Selected Test Cases
P15	[29]	2021	Hybrid Metaheuristic (PSO + Ranked Firefly)	Single objective/multi-objective	Benchmark regression programs	TC Reduction, FC, ET
P16	[30]	2014	Hybrid PSO	Single objective / reduction	Example Software Programs	RR, ET
P17	[31]	2017	GA and BCO	Single objective / multi-objective	Software Test Programs - 17	FA, ET
P18	[9]	2017	Bat Algorithm & Cuckoo Search Algorithm	Single objective /comparative objective	SIR	FC, ET
P19	[32]	2016	ACO	Multi-criteria / prioritization	Automatically Generated Test Scenarios	Test criticality, APFD, ET
P20	[33]	2018	ACO	Single objective / multi-criteria	CBSE Interaction Models	TC Size, Interaction Coverage
P21	[34]	2020	Hybrid HWOA	Multi-objective / cost-oriented	SIR	FD, TC Size
P22	[35]	2016	MFA	Single objective / coverage-oriented	Software Benchmark Programs	Path Coverage
P23	[36]	2017	Improved Cuckoo Search Algorithm	Single objective / cost-oriented	Software Benchmark Programs	APFD, ET, PTR
P24	[37]	2015	Cuckoo Search Algorithm	Single objective / cost-oriented	Sample Software Programs	FD, ET
P25	[38]	2022	Modified ACO	Single objective / multi-objective	Industrial web application	FC, ET, TC
P26	[39]	2023	Multi-Objective Harmony Search	Multi-objective	SIR	AF, ET
P27	[40]	2023	MC-TOA	Multi-objective	Large-Scale Software Systems	HVI, Runtime
P28	[41]	2025	MDEFEA	Multi-objective	Open-Source Software Systems	MCV, WCV
P29	[42]	2025	Select QA	Multi-objective	Open-Source Software Benchmarks for Quantum Computing Systems	Pareto-frontier
P30	[43]	2023	CAOA	Single objective / fault-oriented	A) Five versions of Flex program: flex-v1, flex-v2, flex-v3, flex-v4, flex-v5; B) Four versions of Nano-XML nano-xml-v1, nano-xml-v2, nano-xml-v3, nano-xml-v5; C) Grep; D) Gzip; E) XML Security	DDU

P31	[44]	2023	ABO	Fault-oriented / cost-aware	Sample Experimental Datasets	APFD, ET, TC
P32	[45]	2025	Metaheuristic (Grey Wolf Optimization)	Multi-objective	Open-source benchmark programs	FC, ET, Optimization Accuracy
P33	[46]	2024	AHP-TOPSIS	Multi-criteria decision-making	Technical Debt-Aware Software Testing Scenarios	CC, ET, RF
P34	[47]	2024	Ontology + Multi-Criteria Decision Making	Multi-criteria	Automotive Software Systems	Coverage
P35	[48]	2023	Probabilistic Graphical Model (Belief Propagation)	Probabilistic / cost-aware	Microservice-based systems	ET, FDC
P36	[49]	2022	Learning-based (Recurrent Neural Network)	Learning-based	Neural network models	MAI, FDC
P37	[50]	2020	NSGA-II	Multi-objective	ATL Transformation Zoo (Atlas Transformation Language Repository)	Coverage, ET
P38	[51]	2024	ARTeCS	Multi-objective	XML, RTF, and HL7 Datasets	ET, FDR, TET
P39	[52]	2021	Heuristic-based Selection Strategy	Fault-oriented	Open-source subject programs	FDR, EOE
P40	[53]	2020	Heuristic-based Enhanced Selection Algorithm	Single objective / cost-oriented	Open-source subject programs	ETR, FFDC
P41	[54]	2018	Hybrid Greedy + Evolutionary (GREAP)	Multi-objective	Multi-Version C Programs from Software Infrastructure Repository - 10 Versions	HV, GD, IGD, GS, PFS, Epsilon
P42	[55]	2018	MORTOGA	Multi-objective	SUT	Pareto front, Coverage
P43	[56]	2021	RBCH Algorithm + DBT = Hybrid RBCH-DBT Algorithm	Multi-objective	Software Benchmark Programs	APFD, ET
P44	[57]	2017	Multi-Objective Genetic Algorithm	Multi-objective	Industrial Software Systems	Cost-effectiveness, Diversity
P45	[58]	2024	Hybrid Metaheuristic (BRO + Remora Optimization)	Multi-objective / cost-aware	Benchmark regression programs	FDR, EC, APFD
P46	[59]	2022	Search-Based Optimization (Multi-objective)	Multi-objective	Open-source benchmark programs	Coverage, Similarity Score, ET
P47	[60]	2023	NSGA-II	Multi-objective	Reactor Protection Software Systems	HV, FDC, ET
P48	[61]	2020	Feature-based Heuristic + Optimization	Change-based / feature-oriented	Highly configurable (SPL) systems	FC, FD, EC
P49	[62]	2018	Change-based / Coverage-based Selection	Single objective / criteria-based	Test-driven development projects	RFD, STR
P50	[63]	2019	NSGA-II	Multi-objective	Cyber-Physical and Simulation-Based Software Systems	MHV, ET
P51	[64]	2021	L2-NSGA	Multi-objective	SIR	IGD, HV, IGD
P52	[65]	2020	NSGA-II with Seeding Strategies	Multi-objective	CPS Benchmark Suites	HV, Pareto-frontier
P53	[66]	2016	CPS Product Lines	Multi-objective	Simulation-Based Software Models	FDC, FRC, Pairwise Functional Requirement Coverage, Non-Functional Requirements Coverage, Pairwise Non-Functional Requirement Coverage, ET, Test Suite Similarity
P54	[67]	2020	IOT-CIRTF	Multi-objective/hybrid objective	GSM	CR, FDR, ET
P55	[68]	2025	Hybrid SSA-GA	Learning / prediction-oriented	PROMISE Repository	Accuracy, In-Control MEWMA (IC), Out-of-Control MEWMA (OC), AUC
P56	[69]	2025	MPTCS	Multi-criteria / quality-diversity	RL Experimental Environments	Coverage
P57	[10]	2018	Pareto based Multi-Objective Harmony Search	Multi-objective	SIR	FC, UFC, ET

P58	[70]	2014	Multi-Objective NSGA	Multi-objective	Model-Based Software Systems	Coverage, TC Size
P59	[71]	2021	GA	Single objective / multi-criteria	FSM Models	Mutation Mcore, ET
P60	[72]	2019	Hybrid Heuristic (IR + Coverage)	Hybrid / multi-criteria	Industrial system	APFD, CC, ET
P61	[73]	2022	NSGA-II-MET	Multi-objective	Real Industrial Software System	HV, Pareto-frontier
P62	[74]	2018	Hybrid Similarity-Based Framework	Redundancy-reduction / fault-oriented	Regression testing benchmarks	RRR, FDC