



# An Enhanced Hybrid Artificial Bee Colony Algorithm for Automated Test Case Generation Using Multi-Objective Optimization

<sup>1</sup>Nurudeen Muhammad Bala, <sup>2</sup>Sadiq Umar Abubakar, <sup>3</sup>Musabi Mohammed Shitta and <sup>4</sup>Maryam Muazu Jodi

<sup>1</sup>Centre for Satellite Technology Development, National Space Research Development Agency, Abuja Nigeria

<sup>2</sup>Centre for Satellite Technology Development, National Space Research Development Agency, Abuja Nigeria

<sup>3</sup>Centre for Satellite Technology Development, National Space Research Development Agency Abuja, Nigeria

<sup>4</sup>Dept of computer science: Faculty of Science Education, Shehu Shagari University of Education, Sokoto, Nigeria

Corresponding Author's Email: [nbgusau@gmail.com](mailto:nbgusau@gmail.com)

Received on: August, 2026

Revised and Accepted on: September, 2026

Published on: October, 2026

## ABSTRACT

Automated test case generation plays a vital role in improving software quality by enhancing coverage, reducing human effort, and increasing fault detection efficiency. However, existing approaches face challenges such as: handling multi-objective optimization, maintaining diversity, and avoiding premature convergence. The typical Artificial Bee Colony ABC algorithm is one of many conventional and metaheuristic methods that frequently fail to successfully strike a balance between exploration and exploitation. In order to overcome these constraints, this paper suggests an improved hybrid ABC algorithm for automated test case generation that makes use of a multi-objective optimisation framework. To enhance the quality of the answer, the suggested method combines hybrid search techniques with Pareto-based selection. According to experimental findings, the Hybrid ABC produces a more varied and evenly distributed Pareto front than the traditional ABC in terms of coverage and fault detection potential. The study's findings, hybridisation greatly improves optimisation performance. Future research should concentrate on adaptive hybrid techniques and practical applications.

**Keywords:** Automated Test Case Generation; Software Testing; ABC Algorithm; Hybrid Optimization; Software Quality

## Assurance

### 1.0 INTRODUCTION

Generating high-quality test cases is a crucial component of software engineering, as it greatly impacts software reliability, maintainability, and overall quality. Studies Thota et al. (2020); Matloob et al. (2021) has demonstrated that thorough software testing can uncover approximately 80–90% of defects before system deployment, which in turn greatly lowers maintenance expenses and enhances system reliability. Moreover, Alagarsamy et al. (2024) have commonly noted that software defects found in the latter stages of development are significantly more costly to correct than those discovered during the early testing phases. These insights underscore the necessity for effective and systematic techniques for generating test cases that can optimize coverage and fault detection. However, manual test case production becomes unfeasible, time-consuming, and prone to human error as modern software systems grow more sophisticated, requiring the deployment of intelligent and automated methods.

Automated testing has come a long way, but there are still a number of issues. The inherent challenge between several testing objectives, such as optimising code coverage while concurrently improving fault detection capacity, is one of the main problems. The effectiveness of traditional methodologies in meeting real-world testing needs is limited because they frequently regard test case development as a single-

objective problem (Lakshminarayana & Sureshkumar, 2020). Furthermore, many optimisation algorithms suffer from premature convergence and insufficient diversity in generated solutions, and vast and complicated input domains make exhaustive testing unfeasible. These difficulties frequently result in duplicated or inadequate test cases, which lowers the efficacy and efficiency of testing.

Ahuja & Bhatia, (2023) investigate the use of metaheuristic algorithms for automated test case generation in order to address these problems. Among them, the Artificial Bee Colony (ABC) has been successfully used to a variety of optimisation challenges, including software testing, and was inspired by the foraging behaviour of honey bees (Abbas et al., 2025). Furthermore, other algorithms that have been successful in producing high-quality test cases include Differential Evolution (Santra et al., 2021), Particle Swarm Optimization (Liu et al., 2021) and Genetic Algorithms.

In recent times, hybrid techniques which combine the principles of ABC with various optimisation methods have been introduced to boost convergence rates and solution quality. Nonetheless, a significant number of these approaches continue to lack effective frameworks for managing multi-objective optimisation and typically do not sustain a suitable balance between



exploration and exploitation (Angelov & Lazarova, 2025; Pargaonkar, 2023).

In spite of these improvements, a considerable void continues to exist in the progression of hybrid optimisation frameworks that adequately confront multi-objective test case generation. The research conducted by Akila & Arunachalam, (2022; Chen et al. (2025) typically prioritise the enhancement of individual performance indicators, neglecting the necessary consideration of trade-offs between opposing aims. Additionally, a few studies integrate Pareto-based optimisation strategies with hybrid ABC frameworks to secure convergence alongside a variety of solutions. This gap highlights the need for more advanced approaches that can generate a well-distributed set of optimal test cases while simultaneously optimizing multiple objectives.

To address these issues, this paper suggests an improved Hybrid Artificial Bee Colony method (HybridABC\_ATCG) for multi-objective optimization-based automated test case generation. To successfully balance exploration and exploitation, the suggested method combines Pareto dominance, non-dominated sorting, and hybrid search techniques. The main goal is to produce a varied and excellent collection of Pareto-optimal test cases while optimising simulated coverage and fault detection capabilities.

The remainder of this paper is organized as follows: Section 2 related work. Section 3 describes the proposed HybridABC\_ATCG framework, including problem formulation, algorithm design, and optimization strategy. Section 4 discusses the experimental setup, results, and comparative analysis with the Standard ABC algorithm. Finally, Section 5 concludes the paper and outlines directions for future research.

## 2.0 RELATED STUDIES

According to Alagarsamy et al. (2024); Angelov & Lazarova, (2025) present that many ABC methodologies have been put up for the creation of test cases. For instance, a hybrid ABC technique is suggested by Angelov & Lazarova (2025) however practical scalability and generalisability are questioned. Extremely large effect sizes indicate potential overfitting or overstated performance increases, and results significantly depend on particular parameter settings. Despite being innovative, the fitness function design may skew results in favour of predetermined groups. External validity is weakened by limited comparison with various baseline techniques and real-world datasets, and despite the algorithm's increased complexity, the computational cost implications are not fully explained. Similar to this, Ahmed & Hamdy (2023) how that the Artificial Bee Colony (ABC) algorithm can produce high-

coverage test suites for RESTful APIs using the OpenAPI Specification. However, because it only assesses six APIs, its conclusions are not as broadly applicable. The method prioritises coverage optimisation while ignoring other important goals like test suite minimisation, execution cost, scalability, and the efficacy of fault detection. Its overall advantage is further limited by the lack of comparisons with more contemporary hybrid and multi-objective metaheuristic algorithms.

Furthermore, the enhanced Artificial Bee Colony approach by Ahuja & Bhatia (2023) reduced the test suite by more than 50% with little loss in defect identification; nevertheless, the evaluation was based on a benchmark dataset, which limited proof of practical applicability.

The study ignored other crucial goals including test coverage, execution time, scalability, and computing overhead in favour of minimisation and redundancy elimination. Furthermore, promises of better overall performance were undermined by the exclusion of more modern hybrid and multi-objective optimisation approaches from comparisons. Additionally, the Artificial Bee Colony approach developed by Arasteh et al. (2022) decreased the cost of mutation testing by reducing the number of created mutations; nevertheless, its validation was restricted to benchmark programs and provided little indication of scalability in large industrial software. Faults in less often performed code may be missed if the 80–20 fault-proneness assumption is relied upon. Furthermore, the study did not assess mutation score, fault detection efficacy, or comparisons with modern hybrid and multi-objective optimisation strategies; instead, it concentrated on lowering mutation cost.

Furthermore, the limits of ABC in this application are highlighted by Dhasagounder et al. (2024) show that Cohort Intelligence Optimisation performed better than the Artificial Bee Colony algorithm for PID controller tuning in a two-area interconnected power system. However, the evaluation's generality was limited because it was limited to a particular thermal power system model. Additionally, more current hybrid and multi-objective optimisation methods were not included in comparisons, and real-time implementation, computational cost, and robustness under different operating conditions were not thoroughly examined. Although Lakshminarayana & Sureshkumar (2021) hybrid Cuckoo Search–Bee Colony Algorithm (CSBCA) outperformed PSO, CS, BCA, and Firefly Algorithm in terms of test case optimisation and execution time, the study offers scant proof of its scalability across sizable, real-world software systems.

The evaluation ignored computing overhead, test suite diversity, and the efficacy of fault detection in favour of execution time and path convergence. Additionally, there was a paucity of comparisons with current multi-objective hybrid metaheuristics and statistical significance tests.

### 3.0 METHODOLOGY

For Automated Test Case Generation (ATCG) in a multi-objective optimisation scenario, this paper suggests an improved hybrid optimisation framework based on the Artificial Bee Colony (ABC) method. To effectively provide high-quality test cases, the system combines adaptive search techniques, hybrid learning mechanisms, and Pareto-based optimisation. A multi-objective optimisation problem (MOP) is used to model the automated test case generation problem. Two main objective functions are used to evaluate each test case, which is represented as a candidate solution  $X_i$ :

- I. Objective 1 (Maximise Coverage): Assesses how well a test case covers requirements or program paths.
- II. Objective 2 (Maximise Fault Detection Capability): Calculates the probability of finding defects by minimising their negated value.

The optimization goal is to identify a set of Pareto-optimal solutions, where no solution dominates another based on all objectives.

### 3.1 Experimental Setup Discussion

The purpose of this study's experimental setup is to assess the efficacy of the suggested Hybrid Artificial Bee Colony (ABC) algorithm for automated test case generation in a multi-objective optimisation framework. Each solution represents a potential test case specified by several input variables, and the experiments were carried out using a set of benchmark test case specifications. Maximising simulated test coverage (Objective 1) and maximising fault detection potential (expressed as a negated value in Objective 2 for minimisation purposes) were two competing goals that were taken into consideration. To evaluate gains in solution quality and variety, the Hybrid ABC algorithm's performance was contrasted with that of the Standard ABC algorithm.

To establish comparability, both algorithms were run with the same population size, iteration count, and control settings. A collection of non-dominated solutions that reflected the trade-offs between objectives were captured using the Pareto front technique. In order to calculate average performance metrics and lessen the effect of unpredictability, several runs were carried out. The robustness and excellence of the hybrid strategy in producing optimised and varied test cases were demonstrated by

the assessment metrics, which concentrated on coverage efficiency and fault detection capability.

### 3.2 Problem Formulation and Solution Representation

Each test case is represented as a vector of input parameters and assessed according to goals like maximising coverage and fault detection capability in this study's formulation of automated test case generation as a multi-objective optimisation problem as shown in Eqn (1). Within predetermined constraints, a population of potential solutions is randomly initialised, and each solution is enhanced with properties such as dominated solution sets, objective values, and domination counts. These characteristics make it possible to use Pareto dominance principles, which generate a collection of trade-off solutions by identifying optimal solutions based on non-dominated sorting.

$$X_i = [x_1, x_2, \dots, x_n, \dots, x_N] \dots \dots \dots (1)$$

### 3.3 Multi-Objective Optimization Strategy

Pareto-based ranking is used in the optimisation process to direct the evolution and selection of solutions. The solutions are categorised into hierarchical Pareto fronts and compared using dominance relationships. Fitness assignment ensures a well-distributed set of solutions by combining convergence and diversity. This tactic preserves a wide range of excellent test cases that represent various trade-offs while enabling the algorithm to concurrently optimise competing goals.

### 3.4 Hybrid Artificial Bee Colony Framework

By combining adaptive and guided search methods throughout three phases: employed, observer, and scout bees the suggested HybridABC\_ATCG algorithm improves the traditional Artificial Bee Colony method. The scout bee phase provides random solutions to preserve diversity and avoid stagnation, the employed bee phase uses directed mutation for local exploration, and the observer bee phase probabilistically chooses promising solutions for additional refining. During these stages, multi-objective selection is used to preserve Pareto-superior solutions.

### 3.5 Optimization Process and Evaluation

Population initialisation is the first step in the algorithm's iterative process, which continues with evaluation, non-dominated sorting, and the sequential execution of bee phases until termination criteria are satisfied. The quality of the solution is evaluated using the resulting Pareto front, and convergence and diversity are examined using visualisation tools. Performance is assessed based on the algorithm's capacity to produce a variety of excellent test cases that strike a balance between coverage and fault detection goals. The implementation is done in Python.

#### 4.0 RESULTS

The results of the HybridABC\_ATCG algorithm in Table 1 illustrate a dynamic yet progressively improved search process in creating optimal test cases. Starting with an initial population of 50, the optimal front size changes in the early and mid-iterations, ranging between 2 and 4, which suggests active

exploration of the solution space and avoidance of premature convergence. A dramatic improvement occurs after iteration 60, where the best front size increases significantly, peaking at 7 in iteration 80, demonstrating that the algorithm effectively discovers a richer collection of non-dominated (Pareto-optimal) solutions as it balances exploration and exploitation. Initial population size: 50.

**Table 1: Running HybridABC\_ATCG algorithm**

| S/n | Iteration                         | Best Front Size |
|-----|-----------------------------------|-----------------|
| 0   | Hybrid_ABC_ATCG Iteration 1/100   | 3               |
| 1   | Hybrid_ABC_ATCG Iteration 10/100  | 4               |
| 2   | Hybrid_ABC_ATCG Iteration 20/100  | 3               |
| 3   | Hybrid_ABC_ATCG Iteration 30/100  | 4               |
| 4   | Hybrid_ABC_ATCG Iteration 40/100  | 3               |
| 5   | Hybrid_ABC_ATCG Iteration 50/100  | 2               |
| 6   | Hybrid_ABC_ATCG Iteration 60/100  | 6               |
| 7   | Hybrid_ABC_ATCG Iteration 70/100  | 5               |
| 8   | Hybrid_ABC_ATCG Iteration 80/100  | 7               |
| 9   | Hybrid_ABC_ATCG Iteration 90/100  | 4               |
| 10  | Hybrid_ABC_ATCG Iteration 100/100 | 6               |

The algorithm stabilises with a comparatively high front size of 6 at iteration 100, indicating good convergence and variety, despite minor fluctuations continuing toward the final iterations. Overall, these findings demonstrate how the HybridABC\_ATCG technique can improve multi-objective optimisation performance by preserving a variety of excellent solutions during the search process.

The final Pareto front of the created test cases in the objective space illustrates the trade-off between simulated fault detection potential (negated, also maximised) and simulated coverage (to be maximised), as depicted in Figure 1. A non-dominated solution produced by the HybridABC\_ATCG algorithm is represented by each point on the scatter plot.

The distribution of points shows a distinct trade-off relationship: the negated fault detection potential improves (becomes less negative) as coverage rises from roughly 4.4 to 6.7, showing improved fault detection capabilities. The algorithm successfully finds several optimal solutions rather of converging to a single point, as evidenced by the dispersion of solutions across the front, which shows good diversity. Strong convergence toward the genuine Pareto front, where no option strictly dominates another, is also shown by the points' smooth advance.

Overall, the figure shows that the algorithm successfully strikes a compromise between conflicting goals, giving decision-makers a well-distributed collection of excellent test cases to select from according to particular testing priorities.

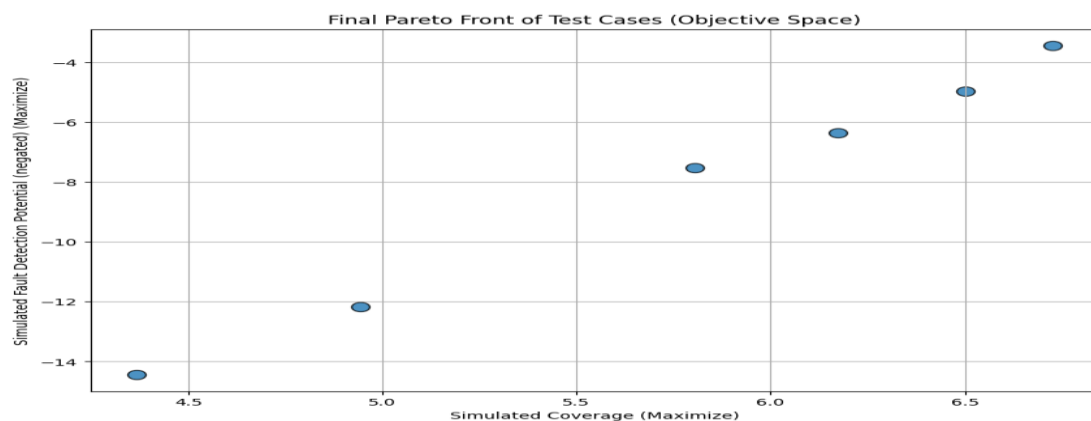


**Figure 1: illustrates the trade-off between simulated fault detection potential**

The solutions derived from the final Pareto front are shown in Table 2, emphasising the trade-offs between the two goals: simulated fault detection potential (Objective 2) and simulated coverage (Objective 1). The findings demonstrate a clear trend in which increased coverage is associated with lower negative (i.e., better) fault detection values. For example, Solution 6 offers the largest coverage (6.73) with somewhat weaker fault detection (-3.44), whereas Solution 1 gets the lowest coverage (4.36) but the

strongest fault detection potential (-14.44). The basic feature of multi-objective optimization that no single solution is always optimal is reflected in this distribution. Rather, each solution provides a distinct balance, enabling testers to choose test cases according to certain criteria, such as improving problem detection or optimising coverage. The HybridABC\_ATCG method efficiently searches the search space to produce a variety of significant alternatives, as evidenced by the diversity of parameter values among solutions.

Table 2: Solutions from the Final Pareto Front

| Solution | Data (Test Case Parameters)    | Objective 1 | Objective 2 |
|----------|--------------------------------|-------------|-------------|
| 1        | [7.78, 8.84, 0.61, 8.86, 1.05] | 4.36        | -14.44      |
| 2        | [0.40, 9.97, 1.83, 2.87, 4.36] | 4.94        | -12.17      |
| 3        | [1.14, 7.42, 1.67, 5.77, 2.73] | 5.80        | -7.52       |
| 4        | [5.59, 5.07, 2.99, 9.99, 3.42] | 6.17        | -6.35       |
| 5        | [4.75, 4.64, 4.95, 0.04, 5.20] | 6.50        | -4.97       |
| 6        | [3.29, 5.00, 1.30, 4.23, 5.13] | 6.73        | -3.44       |

The StandardABC\_ATCG algorithm's performance over 100 iterations is displayed in Table 3, with an emphasis on the size of the optimal Pareto front at various phases. The best front size varied between 2 and 6 during the course of the iterations, according to the results, which show diverse performance. The method occasionally reaches a somewhat high front size (e.g., iterations 1, 20, 50, 70, and 100), but it also drops (most notably at iteration 30 with a front size of 2), indicating inadequate stability in sustaining a

variety of optimal solutions and uneven convergence. This unpredictability suggests that the normal ABC algorithm may find it difficult to consistently strike a balance between exploration and exploitation when compared to the Hybrid method. Overall, the table indicates that although StandardABC\_ATCG can identify good solutions, it lacks the stability and consistent progress shown in more sophisticated hybrid techniques.

Table 3: Running StandardABC ATCG algorithm

| S/N | Iteration                      | Best Front Size |
|-----|--------------------------------|-----------------|
| 1   | Standard ABC Iteration 1/100   | 6               |
| 2   | Standard ABC Iteration 10/100  | 5               |
| 3   | Standard ABC Iteration 20/100  | 6               |
| 4   | Standard ABC Iteration 30/100  | 2               |
| 5   | Standard ABC Iteration 40/100  | 3               |
| 6   | Standard ABC Iteration 50/100  | 6               |
| 7   | Standard ABC Iteration 60/100  | 3               |
| 8   | Standard ABC Iteration 70/100  | 6               |
| 9   | Standard ABC Iteration 80/100  | 3               |
| 10  | Standard ABC Iteration 90/100  | 5               |
| 11  | Standard ABC Iteration 100/100 | 6               |

The distribution of non-dominated solutions produced by both the Hybrid ABC and Standard ABC algorithms in the objective space defined by simulated coverage and negated fault detection potential is shown in Figure 2 of Comparison of Hybrid ABC vs. Standard ABC Pareto Fronts. Better solutions are closer to the ideal Pareto frontier, and the graphic illustrates how each method strikes a balance between increasing coverage and enhancing fault detection capability. The Hybrid ABC's enhanced exploration and exploitation capabilities are demonstrated by a more varied and widely distributed set of solutions throughout the objective space. The Standard ABC Pareto front, on the

other hand, seems less expansive and more concentrated, indicating less diversity and a greater chance of premature convergence.

Additionally, the Hybrid ABC outperforms the Standard ABC in a number of areas, especially when it comes to obtaining superior fault detection values and higher coverage. The hybridisation strategy's ability to improve search performance and preserve solution variety is demonstrated by this better distribution. All things considered, Figure 2 makes it abundantly evident that the Hybrid ABC algorithm offers a more effective and reliable optimisation strategy for multi-objective automated test case development.

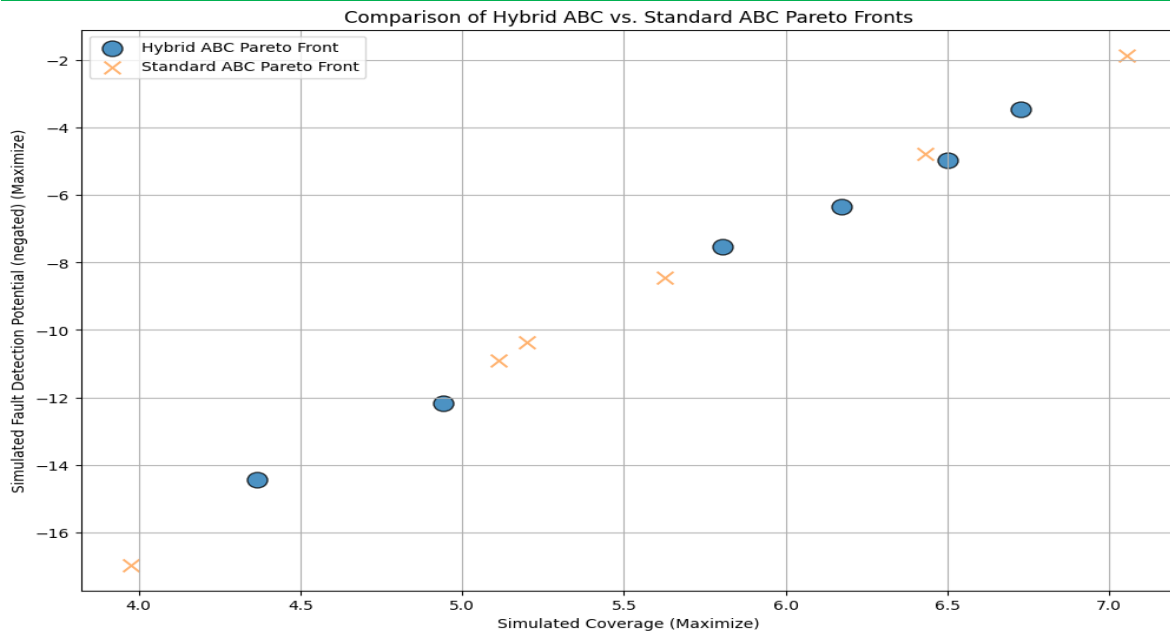
**Figure 2: Comparison of Hybrid ABC vs. Standard ABC Pareto Fronts**

Table 4 presents a better convergence and diversity throughout the trade-off surface are indicated by the Hybrid ABC Pareto front, which is represented by circular markers and shows a more uniform and evenly distributed set of solutions. In comparison to the hybrid technique, the Standard ABC Pareto front (shown with crosses) is less uniform and more dispersed, with certain solutions dominant or less ideal. Particularly in the mid-to-high coverage range, the Hybrid ABC solutions typically attain a more balanced improvement in both objectives, indicating improved optimisation

capabilities. Overall, the figure makes it abundantly evident that the Hybrid ABC algorithm performs better than the Standard ABC in creating a more sophisticated and useful Pareto front for the creation of multi-objective test cases. Solutions from the Final Pareto Front Standard ABC presents the set of non-dominated solutions generated by the Standard ABC algorithm, showing the relationship between simulated coverage (Objective 1) and simulated fault detection potential (Objective 2).

**Table 4: Solutions from the Final Pareto Front Standard ABC**

| Solution | Data (Test Case Parameters)    | Objective 1 | Objective 2 |
|----------|--------------------------------|-------------|-------------|
| 1        | [9.72, 9.95, 3.30, 1.30, 9.64] | 3.98        | -16.97      |
| 2        | [1.08, 7.90, 9.54, 7.44, 7.05] | 5.11        | -10.90      |
| 3        | [1.98, 7.86, 2.89, 8.27, 0.60] | 5.20        | -10.36      |
| 4        | [7.46, 9.11, 6.32, 8.84, 6.69] | 5.63        | -8.45       |
| 5        | [4.88, 5.23, 5.13, 8.27, 8.63] | 6.43        | -4.79       |
| 6        | [4.37, 7.09, 6.80, 5.36, 6.05] | 7.06        | -1.85       |

Table 4 shows a trade-off between the two objectives, with gains in coverage corresponding to less negative fault detection values, as is typical for multi-objective outcomes. For instance, Solution 6 offers the highest coverage (7.06) with lower fault detection efficacy (-1.85), whereas Solution 1 attains the lowest coverage (3.98) but the best fault detection potential (-16.97). In contrast to hybrid approaches, the spread of objective values suggests slightly less balanced optimisation, with some solutions appearing more extreme in one objective than the other. Nevertheless, the variation in test case parameters across solutions shows that the algorithm explores different regions of the search space.

By contrasting the average objective values of the Hybrid ABC and Standard ABC algorithms, the Average Objectives Comparison Table provides an overview of their overall performance. The comparison summary of average objective values for the overall performance of the Standard ABC and Hybrid ABC algorithms is shown in Table 5. In comparison to Standard ABC (5.57), the Hybrid ABC algorithm obtains a higher average simulated coverage (5.75), demonstrating superior efficacy in covering the input space. It also achieves a smaller negative average fault detection potential (-8.15 versus -8.89), indicating enhanced fault detection capacity. These findings imply that, on average, the Hybrid ABC strategy achieves better trade-offs between the objectives and offers a more balanced and effective optimisation

performance. Overall, the data shows how the hybrid approach improves convergence and solution quality when creating multi-objective test cases.

**Table 5: Average Objectives Comparison**

| Index | Algorithm            | Simulated Coverage | Simulated Fault Detection Potential negated |
|-------|----------------------|--------------------|---------------------------------------------|
| 1     | Hybrid ABC Average   | 5.75               | -8.15                                       |
| 2     | Standard ABC Average | 5.57               | -8.89                                       |

## 5.0 CONCLUSION

In order to solve a multi-objective optimisation problem incorporating simulated coverage and fault detection potential, this study evaluated the performance of the Hybrid Artificial Bee Colony (ABC) algorithm with the Standard ABC approach. The average goal values and Pareto front analysis results show that the Hybrid ABC approach consistently achieves a better balance between the two conflicting aims than the Standard ABC. In particular, the Hybrid ABC yields solutions with better (less negative) fault detection potential and higher coverage, indicating more efficient search space exploration and exploitation. Furthermore, the distribution of solutions indicates that the hybridisation improves convergence toward a widely dispersed Pareto front, which eventually results in more varied and superior solutions.

## 6.0 FUTURE WORK

Future research should consider adding more sophisticated hybridisation techniques, including combining machine learning approaches or adaptive parameter tweaking to further improve optimisation efficiency, future study can expand on this work. Deeper understanding of the algorithm's scalability and resilience would also result from extending the issue domain to include higher-dimensional objective functions or real-world software testing scenarios. Additionally, comparisons with other cutting-edge metaheuristic algorithms, like Differential Evolution, Particle Swarm Optimisation, or Genetic Algorithms, could improve the assessment. Lastly, investigating parallel or distributed versions of the Hybrid ABC algorithm and putting dynamic or constraint-based optimisation into practice may increase computing efficiency and applicability to complicated, large-scale issues.

## REFERENCES

- Abbas, N., Eid, H., & Farhat, M. J. (2025). VoI-Aware Energy-Efficient UAV-Assisted Data Collection Using Genetic Algorithms. *Proceedings - IEEE Consumer Communications and Networking Conference, CCNC*, 1–6. <https://doi.org/10.1109/CCNC54725.2025.10976207>
- Ahmed, S., & Hamdy, A. (2023). Automated Black-Box Testing of RESTful APIs Using Enhanced Artificial Bee Colony. *Proceedings - 2023 International Conference on Artificial Intelligence and Data Science (ICAIDS)*, 1–6. <https://doi.org/10.1109/ICAIDS56661.2023.10340522>
- 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>
- Liu, X., Feng, R., Zhou, S., & Yang, Y. (2021). A Novel PSO-SGD with Momentum Algorithm for Medical Image Classification. *Proceedings - 2021 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2021*, 3408–3413. <https://doi.org/10.1109/BIBM49924.2021.9691050>



- <https://doi.org/10.1109/BIBM52615.2021.9669876>  
Matloob, F., Ghazal, T. M., Taleb, N., Aftab, S., Ahmad, M., Khan, M. A., ... & Soomro, T. R. (2021). Software defect prediction using ensemble learning: A systematic literature review. *IEEE Access*, 9, 98754-98771.
- Pargaonkar, S. (2023). A Comprehensive Review of Performance Testing Methodologies and Best Practices: Software Quality Engineering. *International Journal of Science and Research (IJSR)*, 12(8), 2008–2014.  
<https://doi.org/10.21275/sr23822111402>
- Santra, S., Hsieh, J. W., & Lin, C. F. (2021). Gradient Descent Effects on Differential Neural Architecture Search: A Survey. *IEEE Access*, 9, 89602–89618.  
<https://doi.org/10.1109/ACCESS.2021.3090918>
- Thota, M. K., Shajin, F. H., & Rajesh, P. (2020). Survey on software defect prediction techniques. *International Journal of Applied Science and Engineering*, 17(4), 331–344.  
<https://doi.org/https://doi.org/10.6703/IJASE.202012>