UTM UNIVERSITI TEKNOLOGI MALAYSIA
INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING
ISSN 2180-4370
Journal Homepage : https://ijic.utm.my/

An Enhancement of Coverage-based Test Case Prioritization Technique Using Hybrid Genetic Algorithm

Mgbemena Stanley Onyebuchi[1]*, Muhammad Khatibsyarbini[2] & Adham Mohd Isa[3]
Faculty of Computing
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Johor, Malaysia
Email: stanley.buchim@gmail.com[1]; khatibsyarbini@utm.my[2]; mohdadham@utm.my[3]

Submitted: 19/3/2025. Revised edition: 2/5/2025. Accepted: 4/5/2025. Published online: 27/5/2025
DOI: https://doi.org/10.11113/ijic.v15n1.545

Abstract*—Test Case Prioritization plays a defining role in regression testing by optimizing the ordering of test case execution for early fault detection. This ensures that modifications to the code do not adversely affect existing functionalities. TCP is considered an effective approach in regression testing that optimizes test execution by ordering test cases according to a criterion. While previous research works are based on a single criterion using Genetic or Cuckoo search to solve optimization problems in TCP, these single metrics limit their effectiveness. This study proposes a novel hybrid evolutionary algorithm approach that addresses this limitation by significantly improving the Average Percentage Fault Detection (APFD) rates in regression testing. Our adopted approach enhances the strengths of Cuckoo Search Optimization (CSO) to overcome the limitations of premature convergence in GA by parameter tuning helps to overcome the limitations of single algorithm solutions. We evaluate the performance by incorporating APFD and CE for improvement in the rate of fault detection and an increase in coverage effectiveness for a more comprehensive test suite application. Statistical evaluation using ANOVA strengthens our resolve for the adopted approach, with significant results tabulated. Experimental evaluations on Siemens Test Suite datasets demonstrate how our hybrid approach achieves an improvement in APFD of 0.271% over both GA and CSO and an improvement in CE of 6.35% over GA and 6.62%. over CSO across all test suite datasets. These findings highlight the applicability of a well-thought-out hybridized evolutionary algorithm that can be used in TCP to advance software testing practices.

***Keywords*—Test Case Prioritization, Genetic Algorithm, Cuckoo Search Optimization, Hybrid GA-CSO, APFD, CE**

I. INTRODUCTION

Regression testing can be applied on a range of software projects to achieve comprehensive testing. This process is important for developers to prioritize test suites to ensure old functionalities are intact without incurring new bugs [1]. Software quality can be assured by ensuring the software goes through the software testing process [2]. Software plays an undeniable role in today's technologically advanced world when compared to the past two decades, as our lives depend on data for daily increased productivity. These software systems when subjected to modifications are often applied to the lines of code (size) to enhance the abilities or functionalities that can be performed by the software.

The software testing process is both cost intensive and iterative, which can be inhibited partly by time just like any developmental project and this activity is usually conducted at the maintenance phase [3]. The scheduling of test case execution in order of need during regression testing allows for a prioritization approach that increases the testing efficiency of any modified software. Test Case Prioritization processes can be enhanced with some criterion, such that developers can conduct early feasibility study with an expected schedule for project delivery. The level of coverage necessary to achieve a successful test cannot truly be always ascertained, as not all test cases applied reveal faults, besides, not all are utilized [4].

There exist numerous Test Case Prioritization (TCP) strategies, from evolutionary optimization algorithms to machine learning methods that have been proposed to resolve

prioritization problems. Prioritization is then necessary to enhance optimization in TCP as an efficient ordering mechanism for test cases. Studies such as [5] – [8], have found Genetic algorithm (GA) as appropriate to be with beneficial outcomes around TCP. Hence a good bio-metaheuristic algorithm adoption within this study to prioritize test cases.

This study aims to evaluate existing Genetic Algorithm (GA) and Cuckoo Search Optimization (CSO) algorithms for TCP. Average Percentage Fault Detection (APFD) and Coverage Effectiveness (CE) are metrics used in the comparison for evaluation, thereby achieving ordering of test cases. Furthermore, Analysis of Variance (ANOVA) is applied for a statistical representation of the normality of the results generated. The finding will expose future researchers on how to improve and enhance other known evolutionary algorithms towards the progress of TCP. The contributions from this research will broaden the use and application of evolutionary algorithms to TCP. The specific research questions for this study deals with:

i. How to hybridize GA to increase the rate of fault detection in prioritization of test cases in TCP?

ii. How will the hybrid GA be evaluated and verified for efficiency of use in TCP?

II. RELATED WORKS

The concept of prioritizing test cases has become a very effective method applicable in software testing as this process helps place priority on functionality while ensuring that faults are identified faster where necessary. Different research works have proposed varying approaches towards improving the effectiveness of executing test cases.

One of the main objectives of TCP is in the early detection fault rates, during regression testing. Researchers have explored different evolutionary algorithms with a view to improving the effectiveness of TCP. The basic technique for the application of evolutionary algorithms can be achieved with the effective simulation of natural system process to achieve optimization. Study by [9], introduced two search value cognizant TCP approaches Value Cognizant Fault Detection based (VCFDB-TCP) and Value-Cognizant Requirement Coverage-Based TCP (VCRCB-TCP) to improve fault detection and coverage rate of Test Cases considering that they help enhance the business values. The study in [10] introduces a Model-driven Engineering approach TCP to ensure that model input and targeted outputs remain correct after undergoing modification during testing with east of faults detected when tested, while ensuring proper coverage and prioritization of test cases. The study highlighted time-consumption and larger test suite as a limitation for the known TCP approaches and with emphasis to this study a 100% coverage of the test suite was necessary to ensure that the hybridized algorithm achieved more than local optimum needed to avoid easy convergence.

The research by [19] adopted an improved strategy by combining random crossover and dynamic mutation to increase population diversity for the Travelling Salesman Problem using GA. To avoid falling into a localized optimal result, the study ensures that the mutation probability is not limited in size, hence increased the fault detection. This method resulted in an improved convergence rate with better optimal solutions from the improved GA.

Some of the benefits of implementing the code-coverage TCP model are considered effective in its ability to achieve higher average percentages of faults detected, such as block coverage (APBC), statement coverage (APSC), decision coverage (APDC), and statement coverage (APSC). This measure helps testers to quickly identify faults. As one of the most coveted TCP techniques, studies abound in coverage-based approach with single criteria [13], study suggested a modified method for coverage-based criterion by introducing the weighted method for coverage that assesses test cases based on more than one criterion using empirical studies on three standard applications while comparing it to existing methods.

The study expands on the inability for a global solution in TCP while presenting GA and Ant-Colony optimization process to enhance the prioritization process. A metric for the measurement of fault coverage of test cases in a test-suite by considering execution time, and number of iterations towards improving the coverage of a test suite [15].

A. Overview of GA

Genetic Algorithm became popular in the early 1970s when the book by John Holland, “Adaptation in Natural and Artificial Systems” was published, the book was a concept based on the ‘Survival of the fittest’. Genetic algorithms can be categorized under the family of evolutionary algorithms (ET), with their basis on the recognition of superior evolvable candidate solutions referred to as chromosomes. In the ordering of test cases, one identifiable problem is in the representation of the chromosomes. The study presented a permutation encoding solution with a sequence number attached to each test case as a solution [18]. The mutation operation serves as a protection of individual strengths for each identifiable generation.

```
Begin
t ← 0
initialize P(t)
evaluate P(t)
while (not termination condition) do
        begin
        t ← t + 1
        select P(t) from P(t – 1) according to evaluation
        crossover P(t) according to crossover rate
        mutate P(t) according to mutation rate
        evaluate P(t)
end
```

Genetic algorithms in test case prioritization suffer from elitism-based selection which often leads to premature convergence on suboptimal solutions that fail to maximize early fault detection. Also, without proper time-awareness capabilities, GAs struggle to balance execution time with fault detection efficiency, leading to test orderings that achieve coverage inefficiently or detect faults later than necessary.

B. Overview of CSO

A random optimization algorithm to perform swarm search was introduced in 2009 by Yang and Deb, named the cuckoo search algorithm, which is based on the breeding behavior of certain species of cuckoos. The success rate for the cuckoo search as an optimization technique is usually increased by its characteristics and simplicity of implementation fewer parameters [12]. The process of fine-tuning parameter sizes can be a determinant of how to actualize and improve on its performance and exploration ability when realizing the set-out criteria.

The Levy flight strategy, when employed, is considered the best core advantage of CS as this allows for proper exploration of the entire space for a solution. However, it can also be limited in its slow convergence speed and local exploitation ability [16]. The problems of local optimum and low solution qualities exist in optimization; the study adapts a hybrid cuckoo search algorithm to enhance the output [17].

Generate initial population of
n host nests x_i *(i = 1, 2, …, n)*
while *(t < MaxGeneration) or (stop criterion)*
Get a cuckoo randomly by L'evy flights
Evaluate its quality/fitness Fi
Choose a nest among n (say, j*) randomly*
If *(*Fi > Fj*),*
replace j by the new solution;
end
*A fraction (*pa*) of worse nests*
are abandoned and new ones are built;
keep the best solutions
(or nests with quality solutions);
Rank the solutions and find the current best
end while
PostProcess results and visualizations
end

Cuckoo Search Optimization Algorithm for TCP is hindered by the handling of undefined parameters because parameters unpredictability disrupts the search exploration balance, which leads to unreliable convergence to suboptimal test case orderings and lowers adaptability in changing test suites. This makes it difficult to consistently find the most effective test sequence for early fault detection.

This research seeks to solve these limitations of GA and CSO, which can be evaluated by performance metrics such as APFD and CE.

III. METHODOLOGY

This section explains and justifies the research methodology used in this research. It also gives light to the experimental design, experimental context, experimental variables, test case, and evaluation tools used in this research.

A. Justification of the Hybrid Approach

The adopted hybrid technique algorithm has been shown to provide significant potential in addressing the weakness of applying a single objective algorithm. The hybridizing process enhances the search efficiency as GA provides strong selection and crossover function while CSO Levy's light ensures a diversified search, hence reducing premature convergence and improving efficiency.

B. Algorithm of the Hybrid GACSO

The flowchart in Fig. 1 represents the Hybrid Genetic Algorithm Cuckoo Search Algorithm (GACSO). Initially, 50 randomly ranked test suites (nests) from test cases are created. The fitness value of each solution (nest) in the population is computed using the given objective function. The best option is determined by its fitness value.

The Genetic Algorithm Phase begins. In this phase, two parent nests are chosen, and crossover is used to create a child nest. Mutation is then applied to the child nest to create variety. A random nest from the population is selected and its fitness compared to that of the child nest. If the kid nest has a higher fitness level, it replaces the random nest. If the kid nest outperforms the best nest, it replaces it (solution).

The next stage is the Cuckoo Search Optimization (CSO) phase. The fitness value of a cuckoo egg produced in a random nest is assessed. The cuckoo nest takes the place of a randomly chosen nest if its fitness value is higher. The best nest is replaced by the cuckoo nest if it is superior. The poorest 25% of nests in the population are swapped out for new ones if the cuckoo nest does not perform better than these nests. Until the stopping condition, which is 100 iterations, is reached, the entire procedure is repeated recursively. The top-ranked test scenarios were then returned as the optimal solution.

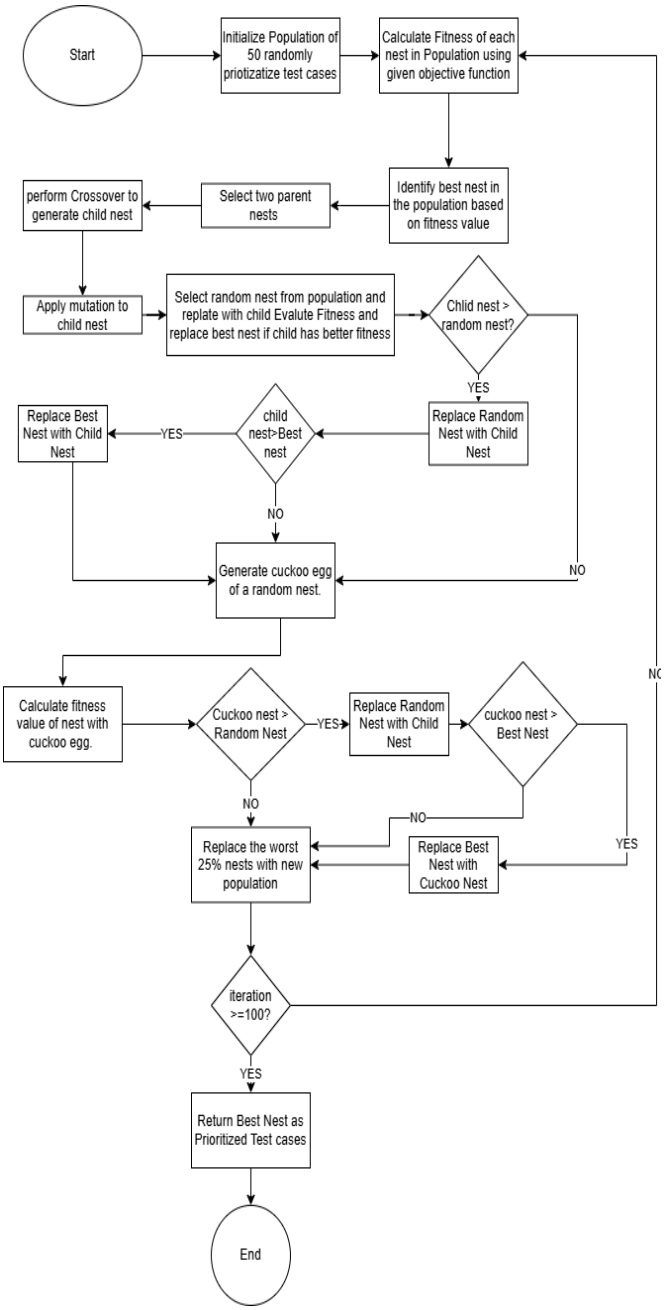

Fig. 1. The Flowchart for GA-CSO

C. Experiment Design

The Fig. 2 shows the conceptual research framework for the adopted methodology.

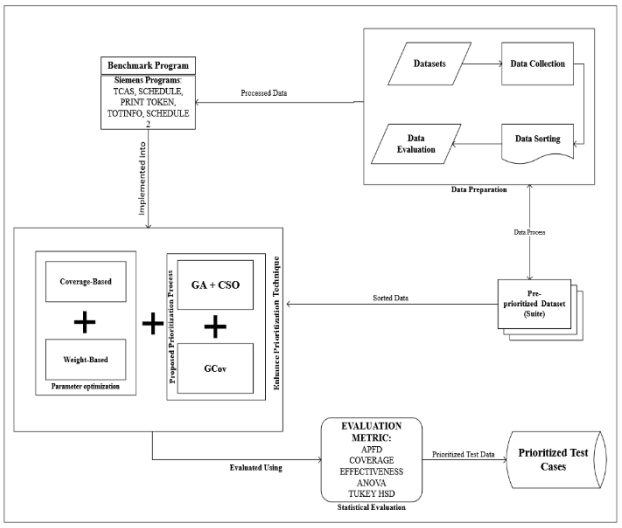

Fig. 2. Research Conceptual Framework

D. Experiment Context

The experiment was conducted in a Linux environment. The software gcov in the Linux environment is used in this context to extract, run, and analyse test cases from every benchmark Siemens dataset. The test case prioritization techniques (GA, CSO, GACSO) are built in Python 3.12 and executed in the Linux terminal. The experiment workstation used had an Intel i7 processor and 16GB of RAM.

E. Experiment Variables

In the context of this experiment and study, we have identified both dependent and independent (controlled) variables for the study. The controlled variables are test case datasets from the Software-artifact Infrastructure Repository (SIR).

The Software-artifact Infrastructure Repository (SIR) is a repository of benchmark software-related artifacts used in software testing techniques. This program was created to study the fault detection capabilities of control-flow and data-flow coverage criteria. It supports controlled experiments in detail with program analysis. The Siemens benchmark program consists of several C programs. The details of the programs are contained in Table II. The dependent variables are APFD and CE.

F. Experiment Test Case

In this experiment, five programs were adopted for the conduct of this experiment and represented in Table I.

TABLE I. APPLIED SIEMENS PROGRAM

Dataset Name	Versions	Test Cases	Lines of Code	Fault Matrix
TCAS	41	1608	138	Different fault for Each version
SCHEDULE	9	2650	299	
SCHEDULE2	10	2710	297	
PRINTTOKENS	7	4130	483	
TOTINFO	23	1053	346	

Table I provides details regarding several software testing benchmark datasets, from the Siemens Program. Five of the seven siemens program where chosen are the (TCAS, SCHEDULE, SCHEDULE2, PRINTTOKENS, and TOTINFO) with their respective metrics. Each program features several versions, with the TCAS program having the highest count at 41 versions and printTokens the least with 7 versions. The quantity of test cases shows considerable variation among programs, with a minimum being 1,053 for Totinfo and the maximum of 4,130 when compared to printTokens. The program constitutes a sizable amount of LOC with printTokens consisting of 483 lines being the largest and the smallest at 138 being Tcas. The fault matrix is indicative of different faults contained within each version.

TABLE II. COMPLETE SIEMENS PROGRAM TEST SUITE

Program	*LoC*	*Task Performed*	*Fault Types*	*Fault Metrics*
schedule	412	Priority Handler	Seeded	Yes
schedule2	374	Priority Handlers	Seeded	Yes
totinfo	565	Statistics Manager	Seeded	Yes
printTokens	726	Lexical Analyzer	Seeded	Yes
printTokens2	570	Lexical Analyzer	Seeded	Yes
replace	564	Pattern Recognition	Seeded	Yes
tcas	173	Collision Detection System	Seeded	Yes

G. Evaluation Tool (Gcov)

Gcov is the coverage analysis infrastructure tool that is available with the gcc compiler and has been in existence since the early 1990 (Blasum *et al*., 2007). Gcov provides coverage analysis tests in which lines of code have been executed and can be utilized in the analysis and verification of the code paths. Coverage is an extensive industrial norm in software engineering as helps in the process of testing the modification of a software.

Gcov consists of three steps that helps define its framework such as the compilation phase, data collection and extraction phase and the reporting phase. The phases are represented by:

- Compilation phase (‘gcc -00 -o hello -fprofile-arcs-ftest-coverage hello c”)
- Data collection and extraction phase (“./hello” is the collecting binary”)
- Reporting phase (‘gcov -a hello c”)

IV. EXPERIMENTS AND RESULTS

The study introduces an evolutionary algorithm adopted for the research, the GA and CSO algorithms' strengths are leveraged, resulting the a hybrid GA-CSO evolutionary algorithm. The proposed hybrid process is then designed to achieve a better rate of fault detection and maximum coverage effectiveness while optimizing execution time.

A. Experiment Motivation

The purpose behind developing an enhanced hybrid evolutionary algorithm comes from identifying the limitations of the individual algorithms. Furthermore, premature convergence, inefficient exploration of the search space, and inability to balance the exploitation and exploration functions effectively. Hence, limiting their ability to effectively provide better solutions in a complex software testing scenario.

B. Experiment Setup

The experimental setup for evaluating the proposed **hybrid GA-CSO** is designed to ensure a fair and comprehensive comparison with baseline algorithms. The Siemens Test Suite from the Software-Artifact Infrastructure Repository (SIR) is used as the benchmark dataset for evaluating test prioritization performance. This suite includes seven programs with seeded faults, enabling an accurate assessment of fault detection capabilities.

To measure the performance, two key evaluation metrics are used. The APFD for fault rate detection and CE for comprehensive coverage of the modified software using SIR test suite. For statistical analysis of the test, we introduced and used ANOVA test for normality and to ensure the statistical significance difference in APFD values. To validate our results, the choice of metrics APFD and CE adopted was limited for this study. If replicated by others, time and cost metrics can be applied when there is availability of resources to measure other outcomes.

C. Initial Results

Tables III and IV and Figs. 3 and 4 represent initial APFD and CE performance results of tests conducted on the GA and CSO algorithms before the hybrid process.

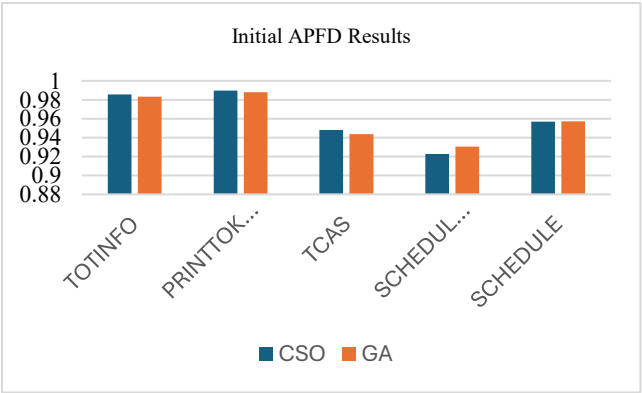

Fig. 3. GA and CSO Initial APFD Chart Result

TABLE III. INITIAL RESULTS APFD RESULTS

APFD Results	CSO	GA
Totinfo	0.98575	0.983431
Printtoken	0.989844	0.988147
Tcas	0.948051	0.943645
Schedule2	0.922633	0.930519
Schedule	0.956994	0.957239

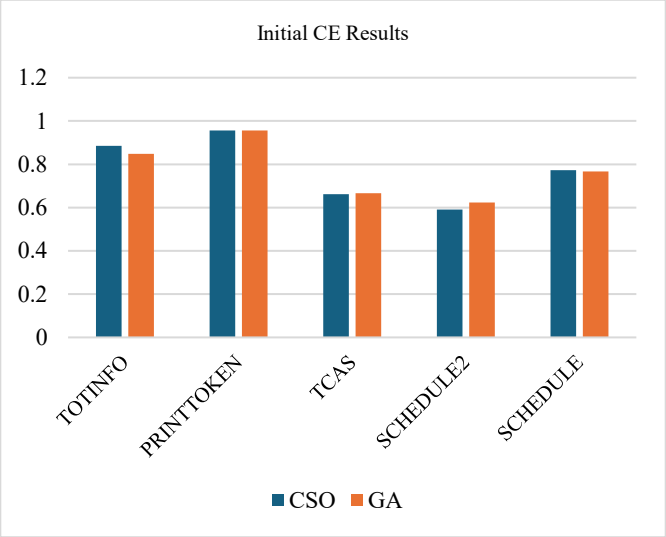

Fig. 4. GA and CSO Initial CE Results

TABLE IV. INITIAL RESULTS CE RESULTS

CE Results	CSO	GA
Totinfo	0.884776	0.848684
Printtoken	0.956066	0.955607
Tcas	0.661899	0.666729
Schedule2	0.590416	0.623637
Schedule	0.77221	0.766996

V. RESULTS AND ANALYSIS

A. Hybridized Results

Post-Hybridization results are contained within Tables V and VI and Figs. 5 and 6:

The comparative analysis of test case prioritization performance across the five benchmark programs reveals consistent superiority of the hybrid GACSO approach over individual GA and CSO implementations.

The Average Percentage of Faults Detection (APFD) results across the five test suite datasets, as shown in Table V and Fig. 5, demonstrate that the hybrid GA-CSO algorithm consistently outperforms both GA and CSO algorithms, though with varying degrees of improvement. For the totinfo test suite dataset, GA-CSO achieves the highest APFD score of 0.9860, showing a slight improvement over CSO (0.9857) and GA (0.9834). While the improvement is modest at 0.026% over CSO and 0.262% over GA. Similarly, for the printToken test suite dataset, the hybrid approach leads with an APFD of 0.9906, compared to CSO at 0.9898 and GA at 0.9881, representing improvements of 0.073% and 0.245% respectively. The Tcas test suite dataset shows more pronounced differences among the algorithms, with GA-CSO achieving an APFD of 0.9497, compared to CSO's 0.9481 and GA's 0.9436. The improvement of 0.177% over CSO and 0.644% over GA. In the schedule test suite dataset, GA-CSO (0.9586) maintains its lead over CSO (0.9570) and GA (0.9572), showing improvements of 0.171% and 0.145% respectively. For the schedule2 test suite dataset, GA-CSO achieves an APFD of 0.9311, compared to CSO's 0.9226 and GA's 0.9305. The improvement over CSO is relatively significant at 0.908%, while the advantage over GA is more modest at 0.059%.

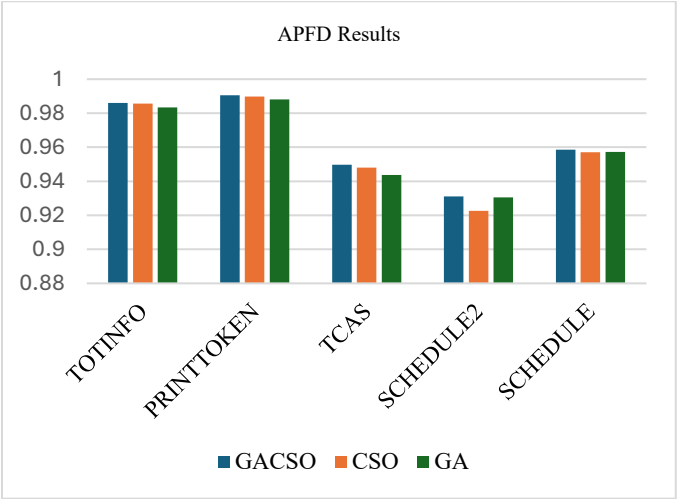

Fig. 5. GACSO, GA and CSO APFD Results

On average, the hybrid GA-CSO algorithm demonstrates an improvement of 0.271% over both GA and CSO across all test suite datasets.

TABLE V. APFD RESULTS FOR GA-CSO, CSO AND GA

APFD Results	GA-CSO	CSO	GA
Totinfo	0.98600781	0.98574994	0.983431
Printtoken	0.990565	0.989844	0.988147
Tcas	0.949727	0.948051	0.943645
Schedule	0.958628	0.956994	0.957239
Schedule2	0.9310638	0.922633	0.930519

The Coverage Effectiveness (CE) results in Table VI and Fig. 6 analysis reveal that the hybrid GA-CSO algorithm consistently outperforms GA and CSO algorithms across all five test suite datasets, showing the effectiveness of combining the genetic algorithm and cuckoo swarm optimization approaches for test case prioritization.

For the totinfo test suite dataset, GA-CSO achieves a Coverage effectiveness value of 0.9641, substantially outperforming both CSO (0.8848) and GA (0.8487). This represents an 8.96% improvement over the CSO and a 13.60% improvement over the GA. The printToken test suite dataset shows all three algorithms perform at the same high level. GACSO still maintains the lead at 0.9566, with modest improvements over CSO (0.9561) and GA (0.9556) at 0.06% and 0.10%, respectively. In the Tcas test suite dataset, GA-CSO achieves a CE of 0.7338 compared to CSO's 0.6619 and GA's 0.6667, representing improvements of 10.86% and 10.06% respectively. Similarly, for the schedule test suite dataset, GA-CSO (0.7890) outperforms both CSO (0.7722) and GA (0.7670), showing improvements of 2.17% and 2.86% respectively. In the schedule2 test suite dataset, we observe that GA-CSO (0.6557) still outperforms both GA (0.6236) and CSO (0.5904), showing an 11.06% improvement over CSO and a 5.15% improvement over GA.

On average, the hybrid GA-CSO algorithm demonstrates substantial enhancements in Coverage Effectiveness, outperforming GA by 6.35% and CSO by 6.62%.

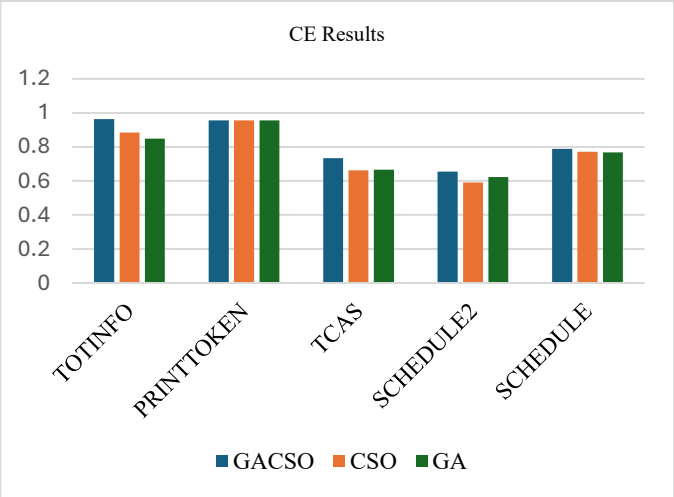

Fig. 6. GACSO, GA and CSO CE Results

TABLE VI. CE RESULTS FOR GA-CSO, CSO AND GA

CE Results	GA-CSO	CSO	GA
Totinfo	0.96409769	0.8847764	0.848684
Printtoken	0.95660609	0.95606582	0.9556074
Tcas	0.733769	0.661899	0.666729
Schedule	0.788965	0.77221	0.766996
Schedule2	0.655739	0.590416	0.623637

These significant improvements highlight the effectiveness of the hybrid approach in maximizing test coverage across diverse test suite dataset types. The results show clearly that the hybrid GA-CSO algorithm successfully combines the exploration capabilities of GA with the exploitation strengths of CSO, resulting in more comprehensive test coverage overall. The consistent superiority of GA-CSO across all test suite datasets, regardless of their complexity or the relative performance of the standalone algorithms, confirms that the hybridization strategy represents a robust advancement in test coverage optimization techniques.

B. Normality Test Using Analysis of Variance (ANOVA)

To assess the normality of the data distributions, Analysis of Variance (ANOVA), also known as the Shapiro-Wilk test, was conducted. The results is shown in Table VII. The null hypothesis ("H_0") of this test states that the data follows a normal distribution, while the alternative hypothesis ("H_1") indicates deviation from normality. A p-value greater than 0.05 suggests that the data is likely to be normally distributed, whereas a p-value less than 0.05 indicates a significant difference from normality.

TABLE VII. THE ANOVA (SHAPIRO-WILK TEST) RESULT FOR THE TEST SUITE

	Shapiro-Wilk	
	df	p-value
totinfo_APFD_GA-CSO	29	0.118
totinfo_APFD_cso	29	0.161
totinfo_APFD_ga	29	0.012
schedule2_APFD_ga	29	0.486
printtoken_APFD_GA-CSO	29	0.707
printtoken_APFD_cso	29	0.047
printtoken_APFD_ga	29	0.001
tcas_APFD_GA-CSO	29	0.076
tcas_APFD_CSO	29	0.516
tcas_APFD_ga	29	0.281
schedule2_APFD_GA-CSO	29	0.408
schedule2_APFD_CSO	29	0.387

C. Interpretation of the Results

The ANOVA (Shapiro-Wilk test) results indicate that the majority of datasets exhibit normality, as their respective p-values exceed the 0.05 significance threshold. Specifically, datasets **totinfo_APFD_GA-CSO (p = 0.118)**, **totinfo_APFD_cso (p = 0.161)**, **schedule2_APFD_ga (p = 0.486)**, **printtoken_APFD_GA-CSO (p = 0.707)**, **tcas_APFD_GA-CSO (p = 0.076)**, **tcas_APFD_CSO (p = 0.516)**, **tcas_APFD_ga (p = 0.281)**, **schedule2_APFD_GA-CSO (p = 0.408)**, and **schedule2_APFD_CSO (p= 0.387)** do not show strong evidence of non-normality.

TABLE VIII. TUKEY HSD RESULT FOR GA-CSO

		Mean Difference	Std Error	p value
GACSO	CSO	0.039930*	0.012634673198570	0.005
	GA	0.058206	0.012634673198570	0.000

The Tukey HSD test aided in identifying value-based difference when multiple test case prioritization techniques are applied to programs such as the Siemens Suite, hence highlighting the statistical significance of the hybridized algorithm in comparison to others. The Tukey HSD results is shown in Table VIII.

D. Summary

The ANOVA (Shapiro-Wilk) tests aids in verification of data distribution patterns to understand its normality. Most of the tested datasets showed normal distribution patterns as their corresponding p-values exceeded 0.05. Data points on printtoken_APFD_CSO, totinfo_APFD and printtoken_APFD_ga showed deviations from normality with p-values 0.047, 0.012 and 0.001 respectively. The chapter uses a well-thought-out analytical framework in our proposed hybrid approach to ensure it performs well in fault detection rates and coverage efficiency, and computational. The statistical test was necessary to highlight comparison done on multiple groups, thereby identifying optimal ranges from search-based algorithms.

VI. CONCLUSION AND FUTURE WORKS

The goal of this study is to identify and conduct a thorough search on the existing body of works and techniques used in TCP, as this will prove useful when identifying the limitations in software testing. To address this limitation, we identified a possible algorithm enhancement technique that hybridizes GA and CSO. In doing so, we were able to increase the APFD rates by enhancing optimization rate and increasing coverage of the test suite using CE as metrics to ensure better than optimum results delivery. Furthermore, the ANOVA and Tukey HSD metrics were applied to confirm statistical outcomes showed normalization in results and performance. This ensures the proposed algorithm can be applied on broader software systems by developers and testers alike. The bias from the dataset selections come from the Siemens programs size do not represent larger industrial scale applications. The parameter tuning performed to successfully conduct the testing might be restrictive to other program languages.

Future work can be focused on the use of real-world projects, improved parameter tuning to accommodate a hybridized GA with other evolutionary algorithms and complex datasets, as well as time monitoring with other APFD metrics for a increased outcomes.

ACKNOWLEDGMENT

The work is fully funded by Fundamental Research Grant Scheme, vote number 23H61 under Universiti Teknologi Malaysia (UTM). We would also like to thank the members of Embedded & Real- Time Software Engineering Laboratory (EReTSEL), Faculty of Computing, UTM for their feedback and continuous support.

CONFLICTS OF INTEREST

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

REFERENCES

[1] Ling, X., Agrawal, R., & Menzies, T. (2022). How Different is Test Case Prioritization for Open and Closed Source Projects? *IEEE Transactions on Software Engineering*, *48*(7), 2526–2540.

[2] Roongruangsuwan, S., & Daengdej, J. (2005). Test Case Prioritization Techniques. *Journal of Theoretical and Applied Information Technology*, 45–60.

[3] Samad, A., Mahdin, H., Kazmi, R., & Ibrahim, R. (2021). Regression Test Case Prioritization: A Systematic Literature Review. *IJACSA) International Journal of Advanced Computer Science and Applications*, *12*(2), 655–663.

[4] Chekam, T. T., Papadakis, M., Le Traon, Y., & Harman, M. (2017). An Empirical Study on Mutation, Statement and Branch Coverage Fault Revelation That Avoids the Unreliable Clean Program Assumption. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 597–608.

[5] Deb, K., Pratab, S., Agarwal, S., & Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NGSA-II. *IEEE Transactions on Evolutionary Computing*, *6*(2), 182–197.

[6] Kaur, A., & Goyal, S. (2011). A Genetic Algorithm for Fault-based Regression Test Case Prioritization. *International Journal of Computer Applications*, *32*(8), 975–8887.

[7] Yuan, F., Bian, Y., Li, Z., & Zhao, R. (2015, September). Epistatic Genetic Algorithm for Test Case Prioritization. In *International Symposium on Search based Software Engineering* (pp. 109–124). Springer, Cham.

[8] Guariso, G., & Sangiorgio, M. (2020). Improving the Performance of Multiobjective Genetic Algorithms: An Elitism-based Approach. *Information 2020, 11*(12), 587.

[9] Ahmed, F. S., Majeed, A., Khan, T. A. (2023). Value-based Test Case Prioritization for Regression Testing using Genetic Algorithms. *Computers, Materials & Continua*, *74(1)*, 2211–2238.

[10] Iqbal, S., & Al-Azzoni, I. (2021). *Test Case Prioritization for Model Transformations*. 6324–6338.

[11] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, (2018). Testcase Prioritization Approaches in Regression Testing: A Systematic Literaturereview. *Inf. Softw. Technol., 93*, 74–93.

[12] Sharma, A., Sharma, A., Chowdary, V., Srivastava, A., & Joshi, P. (2021). Cuckoo Search Algorithm: A Review of Recent Variants and Engineering Applications. *Studies in Computational Intelligence*, *916*, 177–194.

[13] Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2002a). Test Case Prioritization: A Family of Empirical Studies. *IEEE Transactions on Software Engineering*, *28*(2), 159–182.

[14] Prakash. (2013). Weighted Method for Coverage-based Test Case Prioritization. *Journal of Theoretical and Applied Information Technology, 56*(2), 235–243.

[15] Akila, T. K., & Arunachalam, M. (2022). Test Case Prioritization using Modified Genetic Algorithm and Ant

Colony Optimization for Regression Testing. *International Journal of Advanced Technology and Engineering Exploration*, *9*(88), 384–400.

[16] Xiong, Y., Zou, Z., & Cheng, J. (2023). Cuckoo Search Algorithm based on Cloud Model nd Its Application. *Scientific Reports* |, *13*, 10098.

[17] Bajaj, A., & Sangwan, O. P. (2021). Discrete Cuckoo Search Algorithms for Test Case Prioritization. *Applied Soft Computing*, *110*.

[18] Bajaj, A., & Sangwan, O. P. (2019). A Systematic Literature Review of Test Case Prioritization using Genetic Algorithms. *IEEE Access*, *7*, 126355–126375.

[19] Xu, J., Pei, L., & Zhu, R. Z. (2018). Application of a Genetic Algorithm with Random Crossover and Dynamic Mutation on the Travelling Salesman Problem. *Procedia Computer Science*, *131*, 937–945.