electronics

Article

Test Case Prioritization Using Dragon Boat Optimization for Software Quality Testing

Mohammed Assiri

Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Building No 3963, Al-Kharj 16273, Saudi Arabia; m.assiri@psau.edu.sa

Abstract: Test Case Prioritization (TCP) is critical in software quality testing, aiming to identify high-priority test cases early in the testing process. This study proposes a novel TCP approach using the Dragon Boat Optimization Algorithm (DBOA), inspired by the synchronized teamwork seen in dragon boat racing. The proposed TCP-DBOA model strategically reorders test cases to improve fault detection efficiency while minimizing execution time. By using the Average Percentage of Faults Detected (APFD) as the optimization objective, the model enhances both coverage speed and testing effectiveness. DBOA offers advantages in handling large search spaces, balancing exploration and exploitation, and adapting to complex testing scenarios. The performance of TCP-DBOA is evaluated using four benchmark datasets—GZIP, GREP, TCAS, and CS-TCAS—demonstrating superior APFD values compared to existing methods. Results confirm the model's robustness in reducing test execution time and improving fault detection early in the test cycle. This approach contributes to faster, more efficient regression testing, especially in continuous integration environments.

Keywords: Test Case Prioritization; fitness function; software testing; Dragon Boat Optimization Algorithm; software engineering

Academic Editors: Sanghyuk Lee, Manuel Mazzara, Soo Kyun Kim, Scott Uk-Jin Lee, Seokhun Kim and Asad Abbas

Received: 14 February 2025
Revised: 5 April 2025
Accepted: 8 April 2025
Published: 9 April 2025

Citation: Assiri, M. Test Case Prioritization Using Dragon Boat Optimization for Software Quality Testing. *Electronics* **2025**, *14*, 1524. https://doi.org/10.3390/electronics14081524

1. Introduction

Generally, software engineering is software development and programming and the execution of engineering processes in the growth of any software in an organized method [1]. Software testing takes a long time to implement, which is the costliest stage of software development. With the assumption of the agile model in most software enterprises, the interest in constant integration environments is developing [2]. The advantages of such atmospheres include incorporating regular software variations and creating software evolution quicker and less expensive. As an outcome, it will efficiently handle challenges like test implementation, test result reporting, and build procedures [3]. Software testing has been proficient since the area of software engineering arose. Software testing was presented to estimate the quality of the software. Testing contains actions that can classify every possible fault in software for resolving them before the software product has been delivered to end-users [4]. Software testing is frequently implemented, even with time restrictions and fixed resources. The groups of software engineers are commonly required to close their testing actions due to economic and time needs, which will cause some troubles like issues with software excellence and client contracts. However, the TCP app acts to improve test feasibility in software testing action [5]. TCP's primary goal is to fix test cases to attain a preliminary optimizer dependent upon the chosen assets. It provides the capability to perform highly vital test cases as soon as possible according to a few measures and generate preferred outcomes like illuminating previous errors and delivering reviews to the samples. In addition, it aids in addressing a perfect transformation of a sequence of test cases and is implemented consequently. Software testing and debugging costs exceed 50% of the growth cost [6].

The price of regression testing (RT) depends upon the difficulty of the use and the dimension of the test set. The important task of software testing is to enhance the demand of the test cases for execution to identify the highest errors in the assumed test suite. To overcome this task, three common answers are studied in the work, namely test suite prioritization (TSP), test suite minimization (TSM), and test suite selection (TSS). TSP reorganizes the test cases to detect more faults in the highest few test cases. Machine learning (ML) and artificial intelligence (AI) approaches have verified their real suitability in interdisciplinary uses [7]. Finding out the best demand for the test cases and the best method to restrict or pick the test cases creates an NP-hard issue. Optimization approaches are employed to overcome these problems positively. Nature-inspired algorithms have effectively resolved tough optimizer issues in numerous areas. Otherwise, they can enhance the cost-effectiveness of RT. Nature-inspired systems demand researchers due to their simple structure and simplicity of utilization. The techniques have been ideally constructed by exhibiting natural actions [8]. They are generally categorized into biology-inspired, social phenomena-inspired, and physics/chemistry-inspired algorithms. Also, these methods are functional in RT. The most frequently employed algorithms are evolutionary and swarm intelligence-based methods from the biology-inspired family of nature-inspired techniques. The growing complexity of modern software systems demands more effectual testing methods to ensure reliability and quality [9]. Given the time and cost constraints in software testing, optimizing the process has become significant for improving productivity and mitigating expenses. With the rise of continuous development cycles, there is a growing requirement for techniques to prioritize test cases and detect faults early efficiently. This assists in minimizing the overall testing time while ensuring that critical issues are identified promptly. The proposed method aims to address these challenges by introducing a novel approach for TCP, which enhances the overall efficiency and effectiveness of the testing process [10].

This study presents a Test Case Prioritization using the Dragon Boat Optimization Algorithm (TCP-DBOA) model for software quality testing. The chief purpose of the TCP-DBOA model is to diminish the total execution period and maximize the APFD. The DBOA is utilized for TCP. In addition, the TCP-DBOA technique utilizes the average percentage of test point coverage APFD as an optimizer objective to represent the coverage velocity to the test point. The TCP-DBOA technique is recognized as a huge search space for finding an optimum collection of test cases. The performance analysis of the TCP-DBOA method is performed, and the results are investigated in terms of dissimilar measures under diverse datasets.

- The TCP-DBOA model strategically prioritizes test cases to optimize the testing process, effectively mitigating the total execution time. Focusing on an efficient test case order improves overall testing efficiency. This approach ensures faster fault detection and enhanced resource utilization during testing.
- The TCP-DBOA approach aims to optimize the APFD, increasing the chances of identifying faults early in the testing cycle. Prioritizing test cases based on APFD improves fault detection efficiency. This approach results in quicker defect identification, improving the overall efficiency of the testing process.
- The TCP-DBOA methodology implements the DBOA model to improve TCP, allowing the model to navigate large search spaces effectively. By employing DBOA, optimal

test case orders are identified, enhancing testing efficiency. This method ensures a more streamlined process, prioritizing test cases that maximize fault detection.

- The TCP-DBOA technique uniquely utilizes the APFD as an objective function to capture coverage velocity, giving a novel approach to TCP. This method effectively balances fault detection and test case selection (TCS). By optimizing APFD, it ensures a faster and more effective testing process. The novelty is its ability to handle large search spaces while prioritizing fault detection and coverage speed.

2. Related Works

In [11], a model termed TestReduce was developed and planned by a mixture of genetic procedures to discover an enhanced and least pair of test cases. The vital objective of this research is to offer a method that resolves the minimized issue of RT in related desires. The 100-dollar prioritized technique has been employed to describe the significance of novel desires. Hamza et al. [12] intended a Modified Harris Hawks Optimized-based TCP (MHHO-TCP) model for software testing. The main intention of the developed MHHO-TCP method is to expand APFD and diminish the complete implementation period. Furthermore, the MHHO model is intended to increase the exploitation and exploration capacities of the conventional HHO process. Many models have been directed at dissimilar benchmark programs to authorize improved efficacy. In [13], an optimizer algorithm, namely the Bee Algorithm (BA), was projected, dependent upon the intelligent forage performance of the honey bee group. The planned technique advanced for improving the error recognition rate in the least period is stimulated by the performance of dual kinds of worker bees, such as foragers and scout bees. The projected method is executed on dual projects. The prioritized outcome is dignified by employing the APFD. Priya and Prasanna [14] develop an effectual Multi-objective Test Case Generation and Prioritize utilizing an Improved GA (MTCGP-IGA) technique. An arbitrary search-based model for generating and highlighting multiple-objective tests was used. Specifically, the multi-objective optimizer includes increasing the prioritized range of test cases (PR), pairwise coverage of characteristics (PCC), decreasing total implementation cost (TIC), and fault-finding capability (FFC). An exclusive fitness function has been built using cost-effective metrics for the test prioritizing issue.

Iqbal and Al-Azzoni [15] developed an enhanced quantum-behaved particle swarm optimizer (PSO) approach. The model is enhanced using a fix-up device to execute perturbation for the combinatorial TCP issue. Next, the dynamic contraction expansion co-efficient is employed to quicken the convergence. It is surveyed using an adaptive test case collection plan to pick the modification-illuminating test cases. Lastly, the superfluous test cases are detached. In [16], the authors developed a hybrid method for change or RT over the test case prioritized. The recommended model primarily produces the test cases and then groups them into experimental and insignificant clusters by employing a kernel-based fuzzy c-means (FCM) clustering model. Then, the suitable test cases were measured and prioritized by executing the grey wolf optimization (GWO) model. Chandra et al. [17] projected a nature-inspired smell detection agent (SDA) technique. This method is an optimizer algorithm appropriate for recognizing optimum tracks with priority. The SDA procedure depends upon the vanishing of small particles in the gas procedure and the ability of a sensing agent to gain insight. The amount of linearly liberated tracks over a program unit is dignified by generating a control flow graph (CFG), which trials the cyclomatic difficulty. In [18], a test case reduction (TCR) and support-based whale optimization algorithm (SWOA) optimizer for distributed agile software improvement employing RT and including dual phases is projected. Selection and prioritization are implemented once the test cases are rescued and gathered. The test groups have been organized and ranked to confirm that the most dangerous examples were elected foremost. Furthermore, the SWOA has been employed to pick test cases with superior coverage or failure measure occurrence. Table 1 highlights the existing studies on TCP techniques, comparing various approaches in performance metrics such as fault detection, execution time, and efficiency.

Several existing models for optimizing TCP concentrate on improving APFD and mitigating implementation time, but they often face difficulty with scalability for massive datasets. While techniques like BA and MTCGP-IGA aim to improve error detection and coverage, they lack robust mechanisms for adapting to dynamic changes in software environments. Furthermore, approaches such as SDA and SWOA optimizers encounter challenges in generalizing across diverse software applications. Despite improvements, many of these models do not account for the complexities of growing software systems, emphasizing a research gap in developing more flexible, scalable, and adaptive prioritization methods for dynamic testing environments.

Table 1. Existing studies on TCP using optimization algorithm for software quality testing.

Ref. Number	Objective	Methods	Dataset	Measures
Sheikh et al. [11]	To propose the TestReduce technique for minimizing and prioritizing RT cases.	GA	Web application requirements	Test case minimization, prioritization using 100-Dollar approach. Quality criteria conformance evaluation
Hamza et al. [12]	To propose the MHHO-TCP technique for maximizing APFD and minimizing execution time in software testing.	MHHO-based TCP technique	GZIP, GREP, TCAS, and CSTCAS	APFD, ET, FDR
Nayak et al. [13]	To propose a BA-based technique for enhancing fault detection.	BA with Fuzzy Rule Base, Scout And Forager Bees Behavior	Standard Dataset	APFD, FDR, TCP performance
Priya and Prasanna [14]	To propose an efficient MTCGP-IGA for Component-based software development.	Improved GA, Nondominated Sorting GS-II	Component-based Software Development Test Scenarios	TCP, PCC, Fault-Finding Capability (FFC), TIC
Iqbal and Al-Azzoni [15]	To propose a test prioritization approach for the RT model transformations using rule coverage information.	Rule Coverage-based TCP, Empirical Study and Tool Implementation	Model Transformation Test Cases	FDR, TCP Efficiency, Test Case Orderings

Table 1. *Cont.*

Ref. Number	Objective	Methods	Dataset	Measures
Pathik, Pathik, and Sharma [16]	To propose a hybrid technique for RT through TCP using clustering and optimization.	Kernel-based FCM Clustering, GWO for Prioritization	RT Cases for Software Modifications	FDR, TCP Efficiency
Chandra, Sankar, and Anand [17]	To propose a SDA approach for selecting and prioritizing paths in software testing.	SDA, CFG, Cyclomatic Complexity	Ten Benchmarked Applications	Path Coverage Increase, Time Complexity Reduction
Singh, Chauhan, and Popli [18]	To propose a TCR and SWOA for RT in distributed agile software development.	TCP and Selection, SWOA, Clustering and Sorting of Test Cases	Distributed Agile Software Projects	TCS Performance, Coverage and Failure Rate

3. The Proposed Method

This article proposes the TCP-DBOA methodology for software quality testing. The methodology aims to minimize total execution time and maximize the APFD. In addition, it picks the average percentage of test point coverage APFD as an optimizer objective to represent the coverage velocity to the test point. Figure 1 illustrates the overall process of the TCP-DBOA approach.

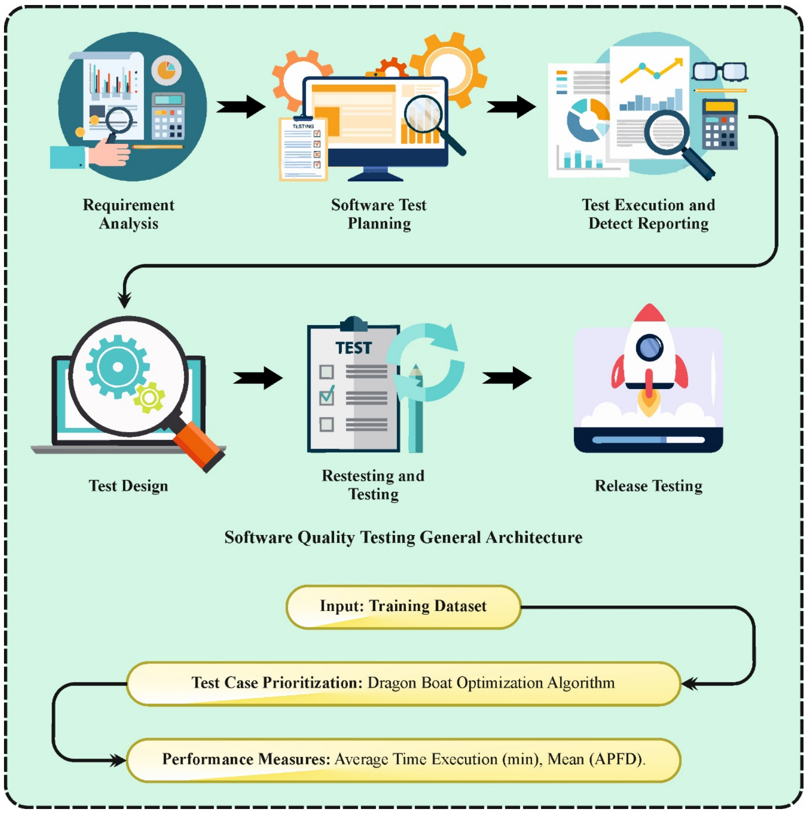
Requirement Analysis
Software Test Planning
Test Execution and Detect Reporting
Test Design
Restesting and Testing
Release Testing
Software Quality Testing General Architecture
Input: Training Dataset
Test Case Prioritization: Dragon Boat Optimization Algorithm
Performance Measures: Average Time Execution (min), Mean (APFD).

Figure 1. Overall procedure of TCP-DBOA approach.

3.1. Design of DBOA

Motivated by the dragon boat (DB) race, the DBOA model examines the conditions of the drummers and paddlers on dissimilar DBs and gathers the racing conditions of all of them [19]. Furthermore, the algorithm integrates a social psychology device to perfect the whole procedure of a DB race. The DBOA model is chosen over other techniques due to its unique capability to combine individual and collective performance through the analogy of a DB race. By considering the roles of drummers and paddlers, the model efficiently captures the dynamics of cooperation and competition within a group, making it highly appropriate for optimization problems. Integrating a social psychology device improves the capability of the model to adapt and refine the racing process, promoting better synchronization and coordination among elements. This aspect allows DBOA to optimize complex tasks more effectively than conventional techniques that may not account for the components' social interaction and collaborative behavior. Additionally, the flexibility and adaptability of the model to various scenarios make it a robust contender for TCP, presenting a more holistic and dynamic approach compared to conventional algorithms. Algorithm 1 demonstrates the DBOA model.

Algorithm 1 DBOA Technique

1. **Initialization:**
   - A population of agents (DBs) is initialized. Every agent has a position and velocity representing a possible solution to the optimization problem.
   - The number of agents depicts the size of the DB team.
   - The optimization problem defines the objective function the algorithm will aim to minimize or maximize.
2. **DB Representation:**
   - The DBs are illustrated by agents with their positions in the search space.
   - Each agent can move across the search space, and their movement depends on the team's synchronization.
3. **Fitness Evaluation:**
   - The fitness function computes how well each DB performs concerning the given objective.
   - The fitness can be based on diverse metrics depending on the problem domain (e.g., APFD, fault detection rate in TCP, etc.).
4. **Synchronization:**
   - Like a DB team where all paddlers synchronize their movements for optimal performance, the agents update their positions by synchronizing with the best-performing members (leaders).
   - The leaders, usually those with the best fitness values, guide the direction of the boat's movement.
5. **Exploration and Exploitation:**
   - Exploration: DBs (agents) explore the search space by slightly adjusting their positions and attempting to discover new regions of the search space.
   - Exploitation: Once a suitable region of the solution space is detected, the agents exploit this by refining their positions to converge towards the optimal solution.

Algorithm 1 *Cont.*

6. **Velocity and Position Update:**
   - The position and velocity of every agent are updated utilizing a formula based on their current position, the best solution found so far, and the team's synchronization. The equation is typically:

     $$\begin{aligned}&\textit{New Position} = \textit{Current Position} + \textit{Velocity}\\ &\textit{New Velocity} = \textit{Current Velocity} + c_1 \times (\textit{BestPosition} - \textit{CurrentPosition}) + c_2 \times (\textit{Global Best Position} - \textit{CurrentPosition})\end{aligned}$$

     where c_1 and c_2 are constants that balance the exploration and exploitation efforts.
7. **Leader Selection:**
   - The agent with the best fitness value becomes the leader, directing the entire team towards better solutions. Other agents adjust their movements based on the leader's position.
8. **Termination Criteria:**
   - The algorithm continues iterating until a stopping condition is met, such as a predefined number of iterations, or if the solution reaches a desired fitness value (convergence).
9. **Final Solution:**
   - After the termination, the best solution (usually the leader's position) is considered the optimal solution to the problem.

3.1.1. Social Behavior Patterns

The DB race is a team sport in which human social powers play a vital part. As an outcome, at the time of race, the teams are subject to the effect of social psychology devices such as social incentives and social loafing. Social loafing denotes when individuals in a group action use less effort than they work alone, often owing to united responsibilities and decreased individual effort levels. Social incentive refers to the application of external factors to inspire creativity, effectiveness, and enthusiasm in groups or individuals when appealing at work or in actions, with the primary goal of enhancing general efficiency.

Considering these social psychology tools, it is thought that when using the DBOA to find an unrestricted issue, the DB team is inclined to display a social loafing performance pattern. However, in the case of constrained problems, the constraining states aid in enhancing the team's inspiration, which results in the behavior of social incentive.

So, the social behavior factor is presented to describe the team's behavior patterns. Its formulation is as follows.

$$\psi = \begin{cases} \frac{DBN}{R_d}, R_d < DBN \textit{ or unconstrained situation} \\ 1,\ R_d > DBN \textit{ or constrained situation} \end{cases} \tag{1}$$

Here, ψ signifies the social behavior factor. DBN represents the number of DBs who participate in a DB race. R_d refers to a random number.

3.1.2. Acceleration Factor

The rate at which the drum beat changes is below the drummer's switch. At the time of the DB race, the drummer analyses the rate at which the drum should be compressed. Calculating this rate takes into consideration many key factors, including the difference in distance enclosed among their own DB and other challenging boats and the state of paddlers. The expression is used to estimate the acceleration factor.

$$\lambda = \frac{\psi \times I - 1}{\psi \times I} \tag{2}$$

Here, λ signifies the acceleration factor. I represents the iteration number.

3.1.3. Attenuation Factor

In a perfect scenario, all paddlers must coordinate their paddling with the drumbeat. However, in actual races, for every crew member, particularly the paddlers, while upholding a high-intensity workout, it is complex to escape the problem of strength attenuation. These outcomes in the paddler's performance worsen as the quantity of paddling rises. The reduction factor is presented to describe this condition. The following expression shows it.

$$\mu = 1 + \frac{\psi \times I - l}{\psi \times I} \tag{3}$$

Here, μ represents the attenuation factor. l designates the number of iterations.

3.1.4. Imbalance Rate of Paddlers

The forward propulsion of a DB is based on the paddler's paddling, which produces a feedback force on the paddle from the sea, pushing the boat forward. To attain the sturdiest forward propulsion, the paddler wants to consider numerous factors before each act of paddling. These factors include the course, distance, angle of the paddling and support created by the paddling, and the degree of obstruction when the boat is moving onward. By following hydrodynamics principles, paddlers can use the water's assets to produce the sturdiest driving force.

Well-executed paddling must uphold an optimum point of entry for the paddle to enlarge the propulsion of the DB, even regarding variations in the water surface. However, when manifold DBs are running similarly, each boat must struggle with its inherent water surface variations and those affected by the paddles of nearby boats. This offers growth at an imbalanced rate, which makes it most challenging for paddlers to maintain steady paddling conditions. The optimum viewpoint at which the paddle arrives at the water can expressively diminish this problem and improve overall performance.

The best paddle angle to entry is θ_B. Here, sinθ is used to compute the paddle's depth in the water, while cosθ helps measure the paddle's water resistance during paddling. The formulation for computing the imbalance rate is as follows.

$$H = \frac{\sqrt{|\cos(\theta)|}}{I * \psi} + H_b \tag{4}$$

Here, H signifies the imbalance rate, portraying the effect of the superposition wave from the surface of the water and the wave produced by the growth of other DBs on the paddler. H_b represents the basis imbalance rate created from the fundamental wave of the water surface. The value of H_b is set to be 0.01 in this paper. θ signifies the entry angle of a paddle.

3.1.5. Strategies for Updating Crew State

The DB's speedy forward movement depends upon the cooperation of the team. The paddlers' condition is represented as a matrix. When upgrading these conditions, the reference object is the condition of the paddler in the equivalent location on the fastest DB. However, a dissimilar upgrade plan is used for the paddlers on the fastest DB. The upgrade plans for paddler conditions are classified into dual cases, such as one for the paddlers on the fastest DB and another for the other DBs. The state upgrade tactics formulations are as follows:

$$Paddlers = \begin{bmatrix} g_1^1 & g_2^1 & \cdots & g_{k-1}^1 & g_k^1 \\ g_1^2 & g_2^2 & \cdots & g_{k-1}^2 & g_k^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ g_1^{j-1} & g_2^{j-1} & \cdots & g_{k-1}^{j-1} & g_k^{j-1} \\ g_1^j & g_2^j & \cdots & g_{k-1}^j & g_k^j \end{bmatrix} \tag{5}$$

$$G_f = Paddlers\ [1,k] = g_k^1, f = 1 \tag{6}$$

$$G_e = Paddlers\ [j,k] = g_{\mathrm{k}}^j, e \neq 1 \tag{7}$$

$$R_f = G_f \times \lambda \tag{8}$$

$$R_e = \frac{c_f + c_e}{2 \times \mu} \times \lambda \tag{9}$$

Here, *Paddlers* signify the state matrix of all paddlers. G_f refers to the paddler's state on the fastest DB. G_e designates the paddler's state on the other DB. R_f represents the state upgrade approach for paddlers on the fastest DB. R_e indicates the other state upgrade strategy for paddlers on the other DB.

3.1.6. Comparative Analysis of DBOA vs. Other Models

The comparative analysis between DBOA and other optimization algorithms, comprising GA, HHO, PSO, differential evolution (DE), shows crucial differences in their performance. DBOA stands out in terms of search space diversity, convergence rate, computational complexity, and success rate. DBOA gives a very high search space diversity compared to other techniques, which assists in preventing premature convergence and ensures that the algorithm explores a broader range of solutions. This feature makes DBOA more effective at escaping local optima. In terms of convergence rate, DBOA is more efficient, reaching optimal solutions faster than GA, which tends to converge more slowly, particularly on larger problem sets. This faster convergence rate is a key advantage in real-world applications where time is critical. Furthermore, DBOA offers low computational complexity, making it more resource-efficient than GA, which usually requires large populations and various generations. As a result, DBOA has a clear advantage in computational cost. Finally, DBOA achieves a very high success rate, outperforming GA and HHO in consistently reaching optimal or near-optimal solutions across various test cases.

The integration of social psychology aspects into DBOA, inspired by the dynamics of a DB race, additionally improves its performance. This teamwork-based approach promotes synchronization and cooperation among agents, improving exploration and exploitation of the solution space. By addressing social behavior patterns such as social loafing and social incentives, DBOA increases the overall team performance, which is substantial for complex optimization tasks. Conventional algorithms, which concentrates primarily on individual optimization, often fall short in handling dynamic and multi-agent problems effectively. Moreover, the capability of the DBOA model to adaptively adjust individual agent efforts based on group dynamics allows for more effectual global exploration, mitigating the risk of stagnation in local optima.

In conclusion, DBOA outperforms GA, HHO, PSO, and DE across diverse critical optimization criteria, making it a more effective and adaptable choice for solving complex, dynamic problems. Table 2 illustrates the performance comparison study of the DBOA with existing methods.

Table 2. Performance comparison of DBOA with GA, HHO, and other approaches.

Algorithm	Search Space Diversity	Convergence Rate	Computational Complexity	Success Rate	Computational Cost
GA	Moderate	Slow	High	Medium	High
HHO	Low	Moderate	Moderate	High	Moderate
PSO	High	Fast	Moderate	High	Moderate
DE	Moderate	Moderate	Moderate	High	Moderate
DBOA	Very High	Fast	Low	Very High	Low

3.2. Process Involved in TCP-DBOA Technique

The proposed TCP-DBOA technique aims to minimize the overall execution time and maximize APFD [3]. The TCP-DBOA technique is chosen for its dual focus on reducing overall execution time while maximizing APFD, which ensures both efficiency and effectiveness in TCP. By mitigating execution time, the model improves testing speed, making it suitable for large-scale projects with tight deadlines. Simultaneously, maximizing APFD ensures higher FDRs early in the testing process, improving software reliability. The TCP-DBOA model strikes a better balance between speed and fault coverage than other techniques, addressing the common trade-off in optimization tasks. Moreover, its capability to handle complex test scenarios and adapt to diverse testing environments gives it a distinct advantage over more conventional models that may prioritize one factor over the other. This makes the TCP-DBOA technique ideal for achieving optimal testing performance in dynamic conditions. Additionally, its flexibility in balancing exploration and exploitation confirms that it can effectively address both simple and highly complex optimization tasks without compromising efficiency.

This study presents a specific example to illustrate the problem of TCP. Assume that there are five test cases and nine components to be enclosed. $T_1 - T_3$ are selected, and all the components are covered as quickly as possible. Compared to the original series of $T_1 - T_2 - T_3 - T_4 - T_5$, the execution sequence of $T_1 - T_3 - T_2 - T_5 - T_4$ is more effective. The tester more quickly addresses the issue in the program. TCP is regularly employed in software testing, which dramatically increases the efficacy of RT. The problem of TCP is determined as the mapping from Pt to the actual number set is F, T denotes the test case, and every possible prioritizing set of test cases in T is Pt. The prioritizing issue of the test case is to locate $T'\varepsilon Pt$ so that for $T''\varepsilon Pt$ and $T' \neq T''$, there is $|f(T') \geq f(T'')|$.

Where f indicates the quantitative report of an objective to measure the performance of prioritization, it is now determined that the better the effect, the larger the f will be. In a real-time application, the tester sets various test objectives, viz., the test point's coverage velocity and the fault's recognition rate. The SI method must determine TCP's fitness function, viz., f values. Based on the requirement, it is classified into single- and multiple-objective optimizers.

The effective execution time (EET) of the test case series shows the time spent by the test case once it obtains the maximal statement coverage for the first time. Moreover, some optimization objectives for code coverage were introduced in the single-objective optimization. For the black box testing, there are almost no mistakes if the elements are enclosed for a significant amount of time. Thus, the study chooses the APTC as an optimizer objective to represent the coverage speed for the testing point. The EET and APTC are mathematically conveyed as follows:

$$EET = \sum_{i=1}^{N'} ET_i \tag{10}$$

$$APTC = 1 - \frac{TT_1 + TT_2 + \cdots + TT_M}{M \cdot N} + \frac{1}{2 \cdot N} \tag{11}$$

where N signifies the test case count, and ET_i indicates the time utilized for implementing i^{th} test case. M shows the program statement counts, N' refers to the count of test cases implemented while the maximal statement coverage was obtained for the primary time, and T_i indicates the test case location from the implementation series where the test point was discovered for the initial time.

4. Result Analysis and Discussion

This section investigates the performance analysis of the TCP-DBOA method. The suggested technique is simulated by employing Python 3.6.5 tool on PC i5-8600k, 250 GB SSD, GeForce 1050 Ti 4 GB, 16 GB RAM, and 1 TB HDD. The parameter settings are provided as follows: learning rate: 0.01, activation: ReLU, epoch count: 50, dropout: 0.5, and batch size: 5.

In Table 3 and Figure 2, the experimental outcomes of the TCP-DBOA method in terms of APFD in terms of the GZIP, GREP, TCAS, and CS-TCAS datasets are given [12]. The results state that the TCP-DBOA technique reaches enhanced values of APFD. With five iterations, the TCP-DBOA technique gains an increased APFD of 97.17%. At the same time, the MHHO-TCP, Fault Analysis (FA), Percentage of Faults Detected (PSD), Location-Based Services (LBS), and greedy approaches obtained decreased APFD of 95.51%, 95.17%, 94.28%, 94.74%, and 92.38%, respectively. Additionally, based on 20 iterations, the TCP-DBOA technique achieves a boosted APFD of 97.13%, but the MHHO-TCP, FA, PSD, LBS, and greedy methods get reduced APFD to 95.59%, 95.01%, 94.56%, 95.13%, and 94.39%, respectively. Meanwhile, with 30 iterations, the TCP-DBOA technique gains an improved APFD of 97.24%, although the MHHO-TCP, FA, PSD, LBS, and greedy techniques acquire diminished APFD of 95.56%, 94.98%, 94.60%, 94.79%, and 93.09%, respectively.

Table 3. APFD analysis of TCP-DBOA technique with various iterations with GZIP dataset.

GZIP Dataset						
Number of Iterations	**TCP-DBOA**	**MHHO-TCP**	**FA Techniques**	**PSD Techniques**	**LBS Techniques**	**Greedy**
1	96.88	95.36	95.15	94.05	94.05	92.39
2	96.59	95.21	94.80	94.19	94.82	92.49
3	96.91	95.31	94.80	94.03	93.87	93.22
4	97.18	95.61	95.37	93.62	95.39	93.10
5	97.17	95.51	95.17	94.28	94.74	92.38
6	97.20	95.56	95.36	93.06	93.78	92.59
7	96.66	95.33	95.15	94.98	95.06	93.56
8	96.49	95.17	94.82	94.35	94.52	93.38
9	97.23	95.59	95.37	94.14	94.62	92.37
10	97.02	95.56	95.33	94.91	95.30	93.29
11	96.69	95.48	94.55	93.89	93.81	93.57
12	96.97	95.64	95.51	93.83	95.06	93.38
13	96.94	95.37	94.83	93.41	93.76	94.24
14	96.76	95.44	95.00	94.55	94.79	93.12
15	96.79	95.56	95.20	93.15	94.59	93.56
16	97.16	95.58	95.28	93.72	93.83	92.29
17	97.07	95.57	95.05	94.03	94.80	93.22
18	97.24	95.86	95.72	94.78	94.43	92.56
19	97.29	95.85	95.71	94.29	94.93	93.53
20	97.13	95.59	95.01	94.56	95.13	94.39
21	97.02	95.69	94.78	94.06	93.93	93.28

Table 3. *Cont.*

GZIP Dataset						
Number of Iterations	TCP-DBOA	MHHO-TCP	FA Techniques	PSD Techniques	LBS Techniques	Greedy
22	96.85	95.59	95.40	93.64	95.42	93.11
23	96.85	95.48	95.21	94.36	94.77	92.41
24	96.97	95.57	95.27	93.12	93.81	92.56
25	97.19	95.49	95.08	95.01	95.02	93.52
26	97.10	95.65	95.52	93.83	95.09	93.34
27	97.27	95.57	94.80	93.40	93.78	94.23
28	97.22	95.74	95.52	93.76	95.11	93.39
29	97.21	95.55	94.82	93.34	93.74	94.26
30	97.24	95.56	94.98	94.60	94.79	93.09

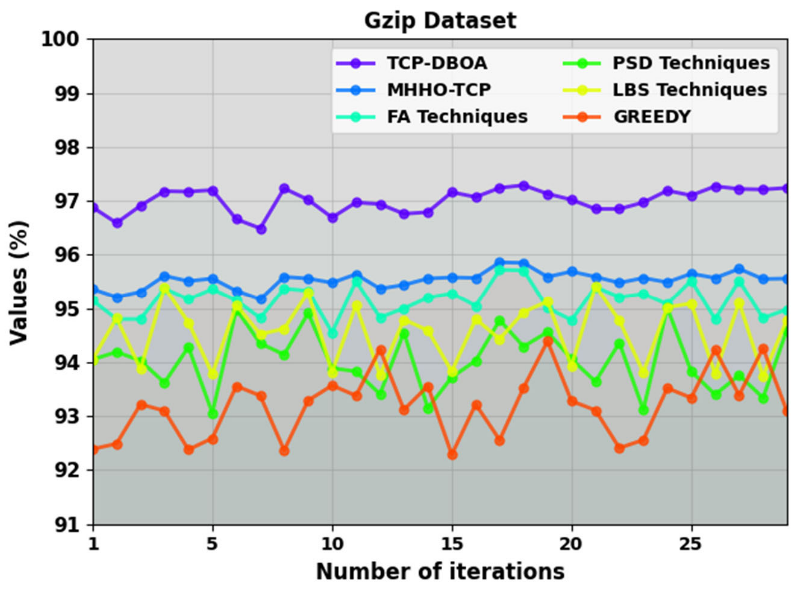

Figure 2. APFD outcome of TCP-DBOA technique on GZIP dataset.

Table 4 and Figure 3 describe the experimental outcomes of the TCP-DBOA method under the GREP dataset with respect to APFD. These acquired outcomes indicated that the TCP-DBOA method obtains improved values of APFD. According to the five iterations, the TCP-DBOA method attains a raised APFD of 97.41%, whereas the MHHO-TCP, FA, PSD, LBS, and greedy methods obtain a diminished APFD of 95.89%, 95.64%, 94.08%, 95.21%, and 93.46%, respectively. In addition, based on 20 iterations, the TCP-DBOA technique provides a higher APFD of 97.33%; however, the MHHO-TCP, FA, PSD, LBS, and greedy methods obtain a lessened APFD of 95.67%, 95.48%, 93.23%, 93.89%, and 92.58%. Likewise, based on 30 iterations, the TCP-DBOA technique achieves a higher APFD of 97.36%, but the MHHO-TCP, FA, PSD, LBS, and greedy methods acquire a reduced APFD of 95.72%, 95.58%, 94.27%, 94.78%, and 92.46%, respectively.

Table 4. APFD analysis of TCP-DBOA technique with diverse iterations on the GREP dataset.

GREP Dataset						
Number of Iterations	TCP-DBOA	MHHO-TCP	FA Techniques	PSD Techniques	LBS Techniques	Greedy
1	96.93	95.63	95.19	94.73	95.31	94.59
2	97.32	96.02	95.90	95.01	94.57	92.60
3	97.40	95.88	95.47	93.93	94.00	92.44

Table 4. *Cont.*

GREP Dataset						
Number of Iterations	TCP-DBOA	MHHO-TCP	FA Techniques	PSD Techniques	LBS Techniques	Greedy
4	97.22	95.66	95.13	94.64	94.97	93.19
5	97.41	95.89	95.64	94.08	95.21	93.46
6	97.12	95.78	95.42	95.05	95.41	93.44
7	97.05	95.54	94.88	93.26	94.72	93.45
8	97.20	95.70	95.44	94.51	93.93	92.68
9	97.31	95.78	95.58	93.74	95.62	93.18
10	96.95	95.64	94.96	94.25	94.98	92.67
11	97.26	95.95	95.82	94.38	95.12	93.66
12	97.11	95.64	95.22	94.17	94.95	93.38
13	97.04	95.61	95.32	93.25	94.78	93.63
14	97.00	95.41	94.93	93.53	93.94	94.33
15	96.92	95.26	94.70	94.04	93.93	93.68
16	97.28	95.70	95.51	94.20	94.74	92.44
17	97.11	95.78	95.20	95.11	95.16	93.65
18	97.28	95.62	95.33	94.47	94.90	92.48
19	97.16	95.54	94.94	94.17	94.01	93.30
20	97.33	95.67	95.48	93.23	93.89	92.58
21	97.24	95.77	95.55	93.74	95.54	93.21
22	96.91	95.53	94.91	94.29	94.98	92.65
23	97.40	95.96	95.86	94.38	95.13	93.66
24	97.38	95.81	95.19	94.19	94.96	93.36
25	97.27	95.73	95.15	94.69	94.96	93.17
26	97.24	95.83	95.64	94.02	95.21	93.49
27	97.15	95.70	95.31	93.25	94.76	93.66
28	97.13	95.51	94.92	93.51	93.87	94.38
29	97.05	95.58	94.68	94.07	94.04	93.71
30	97.36	95.72	95.58	94.27	94.78	92.46

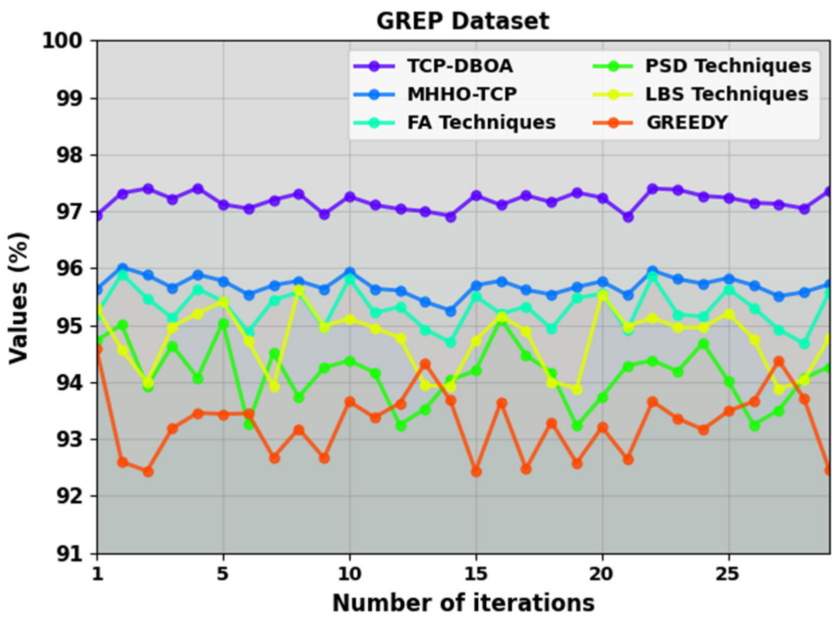

Figure 3. APFD analysis of TCP-DBOA model on GREP dataset.

Table 5 and Figure 4 describe the experimental analysis of the TCP-DBOA technique under the TCAS dataset with respect to APFD. These accomplished findings indicate that the TCP-DBOA technique obtains superior values of APFD. According to 5 iterations, the TCP-DBOA technique provides boosted APFD of 95.67%, although the MHHO-TCP, FA, PSD, LBS, and greedy techniques attain reduced APFD of 94.05%, 94.02%, 94.52%, 92.63%, and 89.81%, respectively. Moreover, based on 20 iterations, the TCP-DBOA method achieves an improved APFD of 95.67%, but the MHHO-TCP, FA, PSD, LBS, and greedy techniques acquire minimized APFD of 94.10%, 94.10%, 92.75%, 94.12%, and 91.62%, respectively. Also, with 30 iterations, the TCP-DBOA method achieves an improved APFD of 95.68%. However, the MHHO-TCP, FA, PSD, LBS, and greedy techniques obtain a reduced APFD of 94.05%, 94.09%, 93.79%, 90.58%, and 92.87%.

Table 5. APFD outcome of TCP-DBOA technique with various iterations under TCAS dataset.

TCAS Dataset						
Number of Iterations	**TCP-DBOA**	**MHHO-TCP**	**FA Techniques**	**PSD Techniques**	**LBS Techniques**	**Greedy**
1	96.35	94.70	94.66	92.43	93.21	91.19
2	95.37	93.77	93.80	93.03	92.42	89.92
3	96.16	94.37	94.35	93.32	93.11	91.70
4	96.69	94.95	94.94	92.81	91.63	91.14
5	95.67	94.05	94.02	94.52	92.63	89.81
6	95.32	93.79	93.80	94.76	90.89	89.77
7	94.35	92.79	92.81	94.64	94.18	90.17
8	96.40	94.76	94.80	91.55	91.34	90.15
9	95.14	93.58	93.61	93.93	93.23	88.52
10	94.97	93.18	93.22	92.19	93.29	91.54
11	96.10	94.42	94.41	93.02	93.33	89.84
12	96.97	95.17	95.21	93.38	91.38	89.55
13	93.87	92.28	92.30	91.41	91.02	92.46
14	95.51	93.97	93.96	93.35	91.49	89.76
15	94.44	92.87	92.87	93.28	89.87	91.50
16	96.79	95.24	95.26	93.57	91.69	90.54
17	96.23	94.46	94.48	94.10	93.60	90.66
18	95.03	93.48	93.45	91.55	92.68	89.42
19	96.07	94.44	94.41	93.78	93.39	92.54
20	95.67	94.10	94.10	92.75	94.12	91.62
21	95.13	93.34	93.34	94.22	92.47	90.59
22	94.89	93.14	93.15	92.83	91.08	90.16
23	94.90	93.17	93.19	94.31	92.24	91.55
24	95.51	93.99	93.99	92.83	91.63	90.02
25	94.74	92.98	93.02	91.62	92.56	90.64
26	96.84	95.25	95.25	94.49	92.45	91.59
27	94.80	93.02	93.00	94.48	93.71	89.48
28	96.89	95.23	95.24	92.13	92.99	90.29
29	95.57	93.79	93.82	93.36	93.82	89.62
30	95.68	94.05	94.09	93.79	90.58	92.87

A wide-ranging experimental analysis of the TCP-DBOA technique in terms of APFD under the CS-TCAS dataset is determined in Table 6 and Figure 5. These achieved outcomes showed that the TCP-DBOA technique obtains increased values of APFD. With five iterations, the TCP-DBOA technique acquires an enhanced APFD of 95.97%, although the MHHO-TCP, FA, PSD, LBS, and greedy approaches obtain a diminished APFD of 94.4%, 94.4%, 94.75%, 91.47%, and 91.18%. Furthermore, based on 20 iterations, the TCP-DBOA method provides a raised APFD of 94.43%. However, the MHHO-TCP, FA, PSD, LBS, and greedy methodologies obtain a reduced APFD of 92.73%, 92.74%, 92.96%, 92.6%, and 93.93%, respectively. In addition, with 30 iterations, the TCP-DBOA method achieves a higher APFD of 95.56%, but the MHHO-TCP, FA, PSD, LBS, and greedy methodologies obtain a lessened APFD of 93.99%, 93.99%, 93.49%, 90.69%, and 92.89%, respectively.

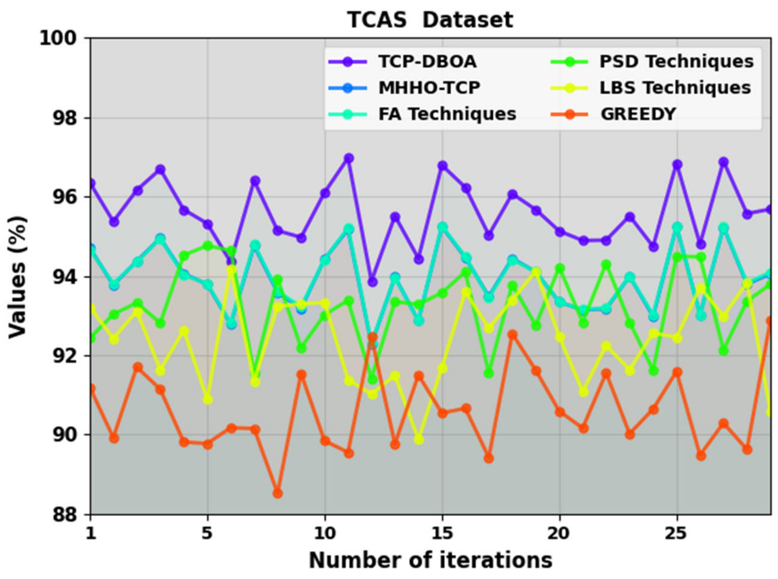

Figure 4. APFD analysis of TCP-DBOA technique under TCAS dataset.

Table 6. APFD analysis of TCP-DBOA technique with varying iterations on CS-TCAS dataset.

CS-TCAS Dataset						
Number of Iterations	TCP-DBOA	MHHO-TCP	FA Techniques	PSD Techniques	LBS Techniques	Greedy
1	96.51	94.79	94.77	92.27	93.33	93.03
2	95.83	94.3	94.29	92.17	93.98	92.26
3	94.88	93.22	93.19	94.27	93.25	93.42
4	95.72	94.06	94.03	94.74	91.75	90.08
5	95.97	94.4	94.4	94.75	91.47	91.18
6	95.46	93.72	93.73	92.15	94.33	90.64
7	94.98	93.18	93.16	92.16	93.52	92.64
8	94.43	92.85	92.86	92.38	93.06	92.03
9	94.99	93.33	93.34	94.39	92.09	90.82
10	95.14	93.57	93.58	92.09	92.23	92.69
11	95.02	93.26	93.25	92.06	91.14	91.13
12	94.62	93.04	93.01	91.97	93.61	92.92
13	94.89	92.23	92.22	93.83	91.93	90.72
14	96.74	95.07	95.07	92.41	94.24	90.29
15	96.08	94.47	94.46	92.94	91.98	89.68
16	96.44	94.73	94.76	93.93	91.38	92.48
17	94.86	93.16	93.15	92.69	91.89	90.9
18	96.04	94.42	94.41	92.69	92.77	91.74
19	96.52	94.76	94.79	94.42	92.37	89.95
20	94.43	92.73	92.74	92.96	92.6	93.93
21	95.97	94.17	94.15	93.15	91.67	92.75
22	95.04	93.33	93.3	92.35	93.56	92.82
23	95.22	93.5	93.47	92.71	94.06	93.55
24	95.27	93.69	93.72	94.04	91.85	90.82
25	96.9	95.1	95.06	94.08	93.27	93.32
26	95.84	94.16	94.15	94.66	94.01	92.55
27	95.15	93.63	93.62	92.76	94.43	91.91

Table 6. *Cont.*

CS-TCAS Dataset						
Number of Iterations	**TCP-DBOA**	**MHHO-TCP**	**FA Techniques**	**PSD Techniques**	**LBS Techniques**	**Greedy**
28	95.19	93.56	93.55	92.55	93.77	91.25
29	95.98	94.2	94.2	93.92	91.07	90.51
30	95.56	93.99	93.99	93.49	90.69	92.89

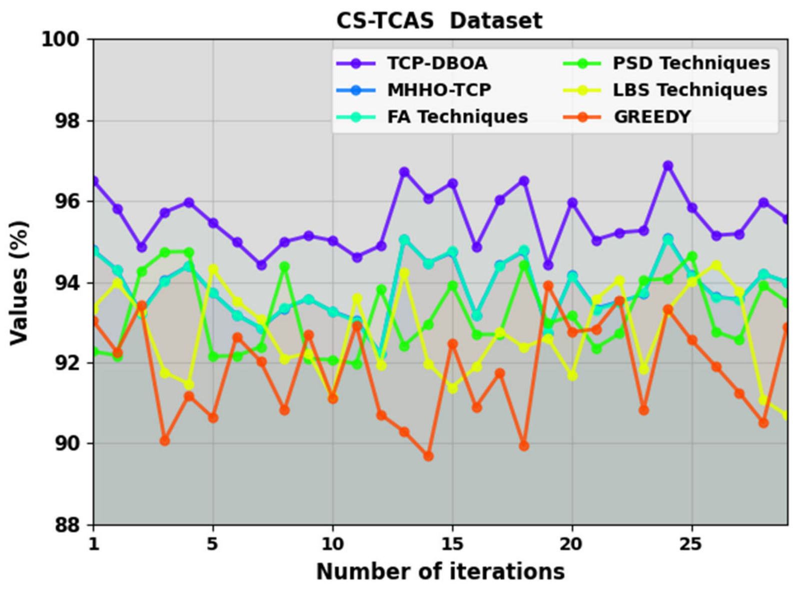

Figure 5. APFD analysis of TCP-DBOA technique with CS-TCAS dataset.

The average time execution (ATE) results of the TCP-DBOA technique are provided with recent models in Table 7 and Figure 6. The results show that the PSD model has a worse performance with maximum ATE values, whereas the FA and Greedy approaches have shown slightly reduced ATE values. Along with that, the MHHO-TCP and LBS techniques obtain reasonable ATE values. However, the TCP-DBOA technique performs better with minimal ATE values of 1.50 min, 1.95 min, 4.69 min, and 7.53 min, under GZIP, GREP, TCAS, and CS-TCAS datasets, respectively.

Table 7. ATE analysis of TCP-DBOA technique with other methods under four datasets.

ATE (min)				
Methods	**GZIP**	**GREP**	**TCAS**	**CS-TCAS**
TCP-DBOA	1.50	1.95	4.69	7.53
MHHO-TCP	3.12	3.75	6.37	9.29
FA Techniques	4.05	4.76	7.67	10.63
PSD Techniques	5.92	6.88	14.38	21.09
LBS Techniques	3.96	4.89	7.61	10.95
Greedy	4.57	4.96	8.73	11.76

The mean APFD results of the TCP-DBOA technique are provided with recent models in Table 8 and Figure 7. The results show that the TCP-DBOA technique gains enhanced mean APFD values. With the GZIP dataset, the TCP-DBOA technique reports an increased mean APFD of 96.92%, while the MHHO-TCP, FA, PSD, LBS, and greedy models obtain a decreased mean APFD of 95.56%, 95.16%, 94.05%, 94.57%, and 93.22%, respectively. Also, based on the GREP dataset, the TCP-DBOA method describes an improved mean APFD of 96.90%, although the MHHO-TCP, FA, PSD, LBS, and greedy techniques achieve a lessened mean APFD of 95.72%, 95.32%, 94.16%, 94.76%, and 93.33%, respectively. Additionally, with the TCAS dataset, the TCP-DBOA method indicates a boosted mean APFD of 94.80%, but the MHHO-TCP, FA, PSD, LBS, and greedy models obtain a reduced mean APFD of 93.65%, 93.12%, 92.40%, 92.07%, and 90.60%. Finally, based on the CS-TCAS dataset, the TCP-DBOA method exhibits an improved mean APFD of 94.80%, although the MHHO-TCP, FA, PSD, LBS, and greedy models obtain a diminished mean APFD of 93.57%, 93.13%, 92.74%, 92.07%, and 91.74%, respectively.

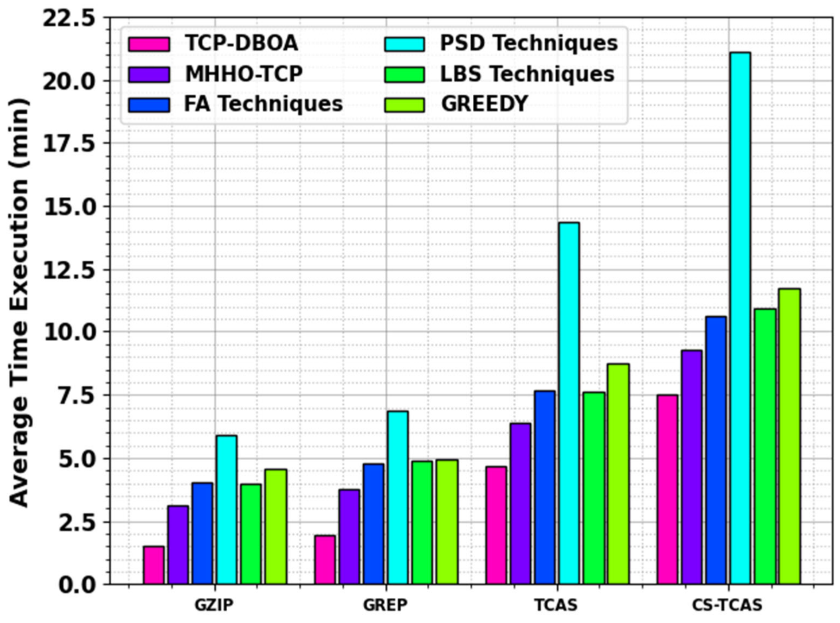

Figure 6. ATE analysis of TCP-DBOA technique under four datasets.

Table 8. Mean APFD analysis of TCP-DBOA technique with other methods under four datasets.

Mean APFD				
Methods	**GZIP**	**GREP**	**TCAS**	**CS-TCAS**
TCP-DBOA	96.92	96.90	94.80	94.80
MHHO-TCP	95.56	95.72	93.65	93.57
FA Techniques	95.16	95.32	93.12	93.13
PSD Techniques	94.05	94.16	92.40	92.74
LBS Techniques	94.57	94.76	92.07	92.07
Greedy	93.22	93.33	90.60	91.74

Table 9 and Figure 8 illustrate the computational time (CT) analysis of the TCP-DBOA approach with existing models. The TCP-DBOA approach demonstrates the most efficient performance across all datasets, with the lowest CT of 7.95 s for GZIP, 6.34 s for GREP, 8.23 s for TCAS, and 6.40 s for CS-TCAS. In comparison, the MHHO-TCP method illustrates significantly higher CT, ranging from 10.97 s for GZIP to 22.83 s for TCAS. The FA, PSD, LBS, and greedy methods also exhibit higher CTs, with the PSD Techniques particularly showing longer CTs, particularly on GREP with 17.85 s and CS-TCAS with 11.72 s. Overall, the TCP-DBOA method provides superior efficiency, completing all tasks faster than the other techniques in all four datasets.

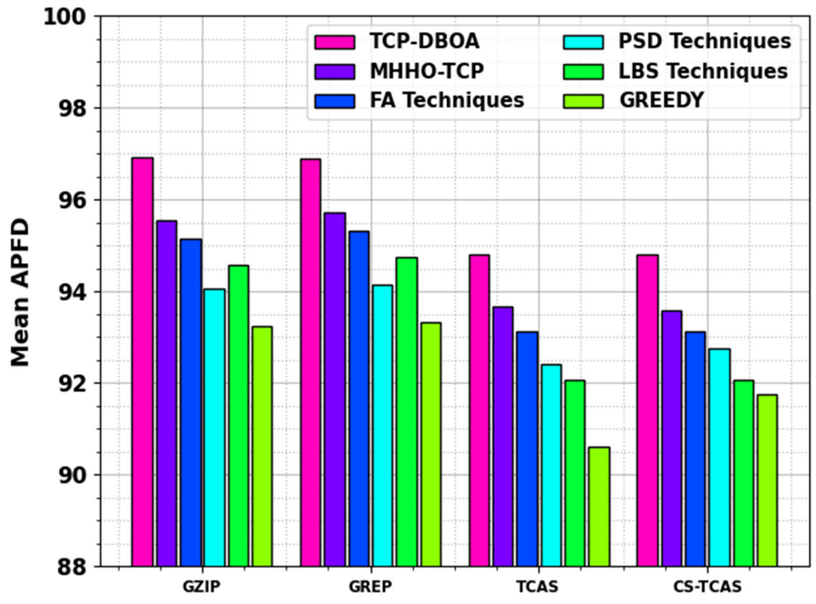

Figure 7. Mean APFD analysis of TCP-DBOA technique under four datasets.

Table 9. CT evaluation of TCP-DBOA technique with existing models under four datasets.

CT (s)				
Methods	**GZIP**	**GREP**	**TCAS**	**CS-TCAS**
TCP-DBOA	7.95	6.34	8.23	6.40
MHHO-TCP	10.97	14.42	22.83	19.88
FA Techniques	13.43	15.73	10.51	22.06
PSD Techniques	23.86	17.85	12.35	11.72
LBS Techniques	19.34	27.21	12.36	23.68
Greedy	11.84	11.61	22.29	11.35

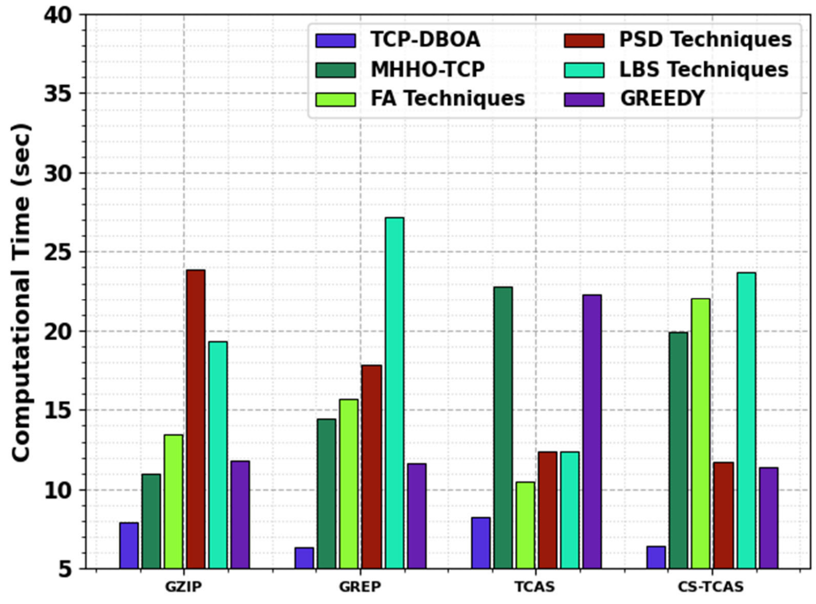

Figure 8. CT evaluation of TCP-DBOA technique with existing models under four datasets.

These results show that the TCP-DBOA technique performs better than recent approaches.

5. Conclusions

In this article, the TCP-DBOA method for software quality testing is proposed. The main purpose of the TCP-DBOA method is to minimize total implementation time and maximize the APFD. In addition, the TCP-DBOA technique picks the average percentage of test point coverage APFD as an optimizer objective to represent the coverage speed to the test point. The TCP-DBOA technique is recognized as a vast search space for finding an optimum organization of test cases. The performance analysis of the TCP-DBOA approach is carried out, and the results are investigated using dissimilar measures. The experimental values highlighted that the TCP-DBOA approach attains better performance over recent approaches. The TCP-DBOA approach's limitations include focusing on a specific set of benchmarks, which may not fully represent the diversity of real-world testing environments. Moreover, the scalability of the approach to handle massive datasets with more complex test cases remains unaddressed. The model also does not incorporate dynamic changes in software or real-time adaptability, which limits its application in agile or continuously evolving development processes. Furthermore, while the study emphasizes fault detection, it does not explore the impact of various environmental factors, such as hardware discrepancies or network conditions, on TCP. Future work may improve the model's scalability, integrate real-time adaptability, and extend its applicability to a wide range of testing scenarios. Further research into hybrid models that incorporate diverse optimization strategies could enhance performance across various software environments.

Funding: This study is supported via funding from Prince Sattam bin Abdulaziz University, project number (PSAU/2025/R/1446).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Nazir, M.; Mehmood, A.; Aslam, W.; Park, Y.; Choi, G.S.; Ashraf, I. A Multi-Goal Particle Swarm Optimizer for Test Case Prioritization. *IEEE Access* **2023**, *11*, 90683–90697. [CrossRef]
2. Silega, N.; Aguilar, G.F.; Alcívar, I.A.; Colombo, K.M. Applying Neutrosophic Iadov Technique for assessing an MDD-based approach to support software design. *Int. J. Neutrosophic Sci. (IJNS)* **2022**, *19*, 80–86. [CrossRef]
3. Yang, B.; Li, H.; Xing, Y.; Zeng, F.; Qian, C.; Shen, Y.; Wang, J. Directed Search Based on Improved Whale Optimization Algorithm for Test Case Prioritization. *Int. J. Comput. Commun. Control* **2023**, *18*, 5049. [CrossRef]
4. Rao, K.K.; Rao, M.B.; Kavitha, C.; Kumari, G.L.; Surekha, Y. Prioritization of Test Cases in Software Testing Using M2 H2 Optimization. *Int. J. Mod. Educ. Comput. Sci.* **2022**, *14*, 56.
5. Li, X.; Yang, Q.; Hong, M.; Pan, C.; Liu, R. Test case prioritization approah based on historical data and multi-objective optimization. *J. Comput. Appl.* **2023**, *43*, 221.
6. Gupta, P.K. K-Step Crossover Method based on Genetic Algorithm for Test Suite Prioritization in Regression Testing. *J. Univers. Comput. Sci.* **2021**, *27*, 170–189. [CrossRef]
7. Juneja, K. Design of a Novel Weighted-Multicriteria Analysis Model for Effective Test Case Prioritization for Network and Robotic Projects. *Wirel. Pers. Commun.* **2022**, *123*, 2505–2532. [CrossRef]
8. Singhal, S.; Jatana, N.; Subahi, A.F.; Gupta, C.; Khalaf, O.I.; Alotaibi, Y. Fault Coverage-Based Test Case Prioritization and Selection Using African Buffalo Optimization. *Comput. Mater. Contin.* **2023**, *74*, 6755–6774. [CrossRef]
9. Raamesh, L.; Jothi, S.; Radhika, S. Test case minimization and prioritization for regression testing using SBLA-based adaboost convolutional neural network. *J. Supercomput.* **2022**, *78*, 18379–18403. [CrossRef]
10. Rajagopal, M.; Sivasakthivel, R.; Loganathan, K.; Sarris, L.E. An Automated Path-Focused Test Case Generation with Dynamic Parameterization Using Adaptive Genetic Algorithm (AGA) for Structural Program Testing. *Information* **2023**, *14*, 166. [CrossRef]
11. Sheikh, R.; Babar, M.I.; Butt, R.; Abdelmaboud, A.; Eisa, T.A.E. An Optimized Test Case Minimization Technique Using Genetic Algorithm for Regression Testing. *Comput. Mater. Contin.* **2023**, *74*, 6789–6806. [CrossRef]
12. Hamza, M.A.; Abdelmaboud, A.; Larabi-Marie-Sainte, S.; Alshahrani, H.M.; Al Duhayyim, M.; Ibrahim, H.A.; Rizwanullah, M.; Yaseen, I. Modified Harris hawks optimization based Test Case Prioritization for software testing. *CMC-Comput. Mater. Contin.* **2022**, *72*, 1951–1965.

13. Nayak, S.; Kumar, C.; Tripathi, S.; Mohanty, N.; Baral, V. Regression test optimization and prioritization using Honey Bee optimization algorithm with fuzzy rule base. *Soft Comput.* **2021**, *25*, 9925–9942. [CrossRef]
14. Priya, T.; Prasanna, M. Component-Based Test Case Generation and Prioritization Using an Improved Genetic Algorithm. *Int. J. Coop. Inf. Syst.* **2023**, *34*, 2350017. [CrossRef]
15. Iqbal, S.; Al-Azzoni, I. Test Case Prioritization for model transformations. *J. King Saud Univ. -Comput. Inf. Sci.* **2022**, *34*, 6324–6338. [CrossRef]
16. Pathik, B.; Pathik, N.; Sharma, M. Test Case Prioritization for changed code using nature inspired optimizer. *J. Intell. Fuzzy Syst.* **2023**, *44*, 5711–5718. [CrossRef]
17. Chandra, S.V.; Sankar, S.S.; Anand, H.S. Smell Detection Agent Optimization Approach to Path Generation in Automated Software Testing. *J. Electron. Test.* **2022**, *38*, 623–636. [CrossRef]
18. Singh, M.; Chauhan, N.; Popli, R. Test Case Reduction and SWOA Optimization for Distributed Agile Software Development Using Regression Testing. *Multimed. Tools Appl.* **2023**, *84*, 7065–7090. [CrossRef]
19. Li, X.; Lan, L.; Lahza, H.; Yang, S.; Wang, S.; Yang, W.; Liu, H.; Zhang, Y. A Novel Human-Based Meta-Heuristic Algorithm: Dragon Boat Optimization. *arXiv* **2023**, arXiv:2311.15539.