International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
©2022 INPRESSCO®, All Rights Reserved
Available at http://inpressco.com/category/ijcet

Research Article

# Optimizing Program Path Coverage in Big Data Software Testing through Divide and Conquer and Hybrid PSO-Simulated Annealing Techniques

**[1]*Sathiyendran Ganesan, [2]Nagendra Kumar Musham, [3]Venkata Sivakumar Musam and [4]Aravindhan Kurunthachalam**

[1]Troy, Michigan, USA
[2]Celer Systems Inc, California, USA
[3]Astute Solutions LLC, California, USA
[4]SNS College of Technology, Coimbatore, Tamil Nadu, India.

Received 10 Jan 2022, Accepted 10 Feb 2022, Available online 15 Feb 2022, **Vol.12, No.1 (Jan/Feb 2022)**

***Abstract***

*Software testing is important to make sure that applications are reliable, functional, and secure. It involves many aspects, and test case generation and program path coverage are considered the most vital because it increases fault detection by covering all possible execution paths. Many traditional approaches like Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) have been employed in test case generation. Unfortunately, these approaches do not perform well with larger, complex software systems due to the high cost of computations, getting stuck in local optima, and really poor scalability. In order to tackle the mentioned limitations, this research thus proposes an optimized test path generation framework according to hybrid PSO-SA approach: It combines static analysis with Control Flow Graph (CFG) and Program Dependency Graph (PDG) and is followed by feasible execution paths extraction from the static analysis. A Divide and Conquer-based clustering technique applied then clusters the paths based on similarity. The next step after the clustering is the optimization performed using Hybrid PSO-SA algorithm. Optimized paths are bundled together to eliminate almost all forms of redundancy while maximizing path coverage. The evaluations clearly indicate the proposed approach attaining superior test coverage, efficiency, and fault demarcation than existing approaches. It saves on computational overhead while improving the effectiveness of the test suite through the elimination of redundant test paths. The findings show that the Hybrid PSO-SA method can be a very good method for increasing software testing efficiency, especially for large-scale applications that find it hard to scale and adapt with traditional methods.*

***Keywords:*** *Software Testing, Test Path Optimization, Particle Swarm Optimization, Simulated Annealing, Path Coverage.*

## 1. Introduction

Software testing has acquired its inevitable place in the software engineering lifecycle as the most crucial activity to ensure the reliability, functionality, and security aspects of software applications [1]. It plays a vital role in delivering high-quality software that meets user requirements and expectations [2]. Software testing has many purposes, out of which one of the most important is finding defects and eliminating them to ensure a successful and quality product from the vendor's perspective [3]. Test case generation is a fundamental part of the software testing process, designed to systematically create input sets for testing various software functions [4].

Program path coverage is another critical component, which involves generating test cases representing all possible execution paths within a program [5]. High path coverage ensures that the largest possible portion of the software logic, including branches and decision points, is tested [6]. This comprehensive testing reduces the likelihood of bugs going undetected and increases overall software robustness [7]. Effective test case generation aimed at maximizing path coverage becomes especially important in complex and large systems, where the number of possible paths can be enormous [8]. In such cases, test coverage directly influences the quality and thoroughness of the testing process, impacting software reliability [9].

In recent years, several techniques have been proposed to automate and improve test case generation and path coverage [10]. Evolutionary

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.12.1.4

algorithms such as genetic algorithms (GA), particle swarm optimization (PSO), and simulated annealing have been widely explored in this context [11]. These approaches aim to efficiently search the input space to generate diverse test cases with high coverage [12]. While these methods have demonstrated promising results in many scenarios, they face significant challenges when applied to large-scale and Big Data software systems [13]. Among the major problems are high computational costs that limit their scalability [14]. They are also prone to getting trapped in local optima, which reduces the diversity and effectiveness of generated test cases [15]. Moreover, the rapid growth and complexity of Big Data systems introduce dynamic changes in software execution paths that many traditional techniques fail to consider [16]. The dynamic nature of Big Data environments can cause frequent alterations in program paths, increasing the difficulty of achieving effective path coverage [17]. As a result, conventional optimization methods may not be sufficient for real-world Big Data applications [18].

This study addresses these issues by proposing advanced optimization techniques to enhance program path coverage in Big Data software testing [19]. The goal is to generate more efficient and varied test cases capable of covering the most complex execution paths in very large software systems [20]. The research focuses on improving the path coverage process specifically for Big Data contexts, where traditional methods typically underperform [21]. The study also includes systematic software decomposition strategies to break down large systems into manageable components for more effective testing [22]. This combination aims to maximize test coverage while ensuring the reliability and quality of software testing processes [23]. Ultimately, the proposed approach targets modern, large-scale applications by enhancing the thoroughness and effectiveness of software testing in challenging Big Data environments [24]. By doing so, it contributes significantly to advancing the state-of-the-art in software testing methodologies [25].

**Research Contributions**

- Challenges in traditional test case generation such as computer overhead and scalability are analyzed.
- Making hybrid PSO-SA frameworks for optimizing test paths and improving coverage.
- Performance comparisons in evaluating the approach to its efficiency and reliability.

## 2. Literature Survey

NOMA, UVFA, and DGNNs are the techniques that resources are allocated dynamically, approximating functions in the efficient generation of test cases and program path coverage in software testing [26]. RFE + ELM + SRC may find it difficult to achieve dynamic path coverage in software testing as it uses static feature selection and thereby lacks the adaptability required in evolving test case scenarios [27]. RPMA is integrated with BLE and LTE-M and GMM to control IoT devices, using Gaussian Mixture Models (GMM) to enable smart resource allocation and intrusion detection [28]. In software testing, particularly in regression testing, the study proposes combining neural networks with heuristic approaches as hybrid techniques to originate in a very unique way Critical Tectonophysics Test Case Prioritization (TCP) [29]. This research article contributes to highlights over a number of technologies such as DROP, AES encryption, and neural networks in AI-driven CAPTCHAs and graphical passwords, yet with a disadvantage of computational overhead and path coverage issues in software testing [30]. Automated fault injection and XML-based test case generation should not be isolated from the liabilities of complex path coverage in very-sized systems along with challenges while updating fault libraries for effective test cases generated [31]. Any deep learning techniques also share several drawbacks like lack of interpretability and specific dataset diversity that can be attributed to issues like unclear path coverage and insufficient test case generation found in software testing [32].

This demonstrates that AWS Fault Injection Simulator (FIS) and sophisticated fault injection techniques would be used in resiliency tests for AWS-based applications, proactive fault injection, and real-time monitoring [33]. Digital economy-cum-entrepreneurial growth was studied with a Threshold Regression Model to establish the effect of the digital economy on industrial structure upgrading [34]. The hybrid language model coupled with evolutionary algorithm approach has the tendency of very high computational cost and difficulty in handling dynamic program paths in test case generation [35]. The research shows that KNN, LPQ, and Bayesian optimized MLP are used for tumor detection. Another such approach may yield layered test case generation for sophisticated software path evaluation [36]. The research discussed involves the dynamic use of metadata reconstruction in the form of multibiometric key generation for the secured storage and transfer of patient data in mobile healthcare systems with cloud computing integration [37]. However, ANN + FEA + Electro-thermal modelling requires relatively high computation and may have difficulties covering real-time paths due to the complexity of behaviours in systems [38].

This study infers the combination of Adaptive Gradient Support Vector Regression (AGSVR), Long Short-Term Memory (LSTM), and Hidden Markov Models (HMM) for detection of malware. Such methods support analysing data sequences for discovering malicious patterns [39]. Challenges summarized under privacy issues and regulatory requirements and model adaptation also hamper the development of this framework in AI and data analytics which tend to affect test case generation and path coverage precision in software testing [40]. Research integrating multiobjective optimization and neural symbolic machines indicates a significant potential in enhancing adaptive test case generation, but the computational overhead and model complexity limit real-world applications [41]. Novel reinforcement learning algorithms combined with symbolic execution are being explored for automated test path discovery to address coverage gaps [42]. Recent advancements in graph neural networks (GNNs) enable better modelling of program control flow for optimized test data generation [43]. Ensemble learning techniques alongside metaheuristic optimization have been applied to prioritize test cases dynamically, improving testing efficiency in cloud-based environments [44]. Lastly, the application of federated learning in distributed software testing environments presents promising results for privacy preservation and scalability in path coverage analysis [45].

Advances in explainable AI have been integrated into testing frameworks to improve model interpretability while maintaining accuracy in path coverage assessment [46]. Adaptive metaheuristic algorithms such as ant colony optimization and genetic algorithms have been applied to optimize test case prioritization, reducing computation time while enhancing coverage [47]. Deep reinforcement learning approaches have been used for dynamic test case generation in continuous integration pipelines, showing improved adaptability to code changes [48]. Transfer learning has been explored to leverage pre-trained models for effective test data generation in resource-constrained environments [49]. Research also shows that the combination of evolutionary computation and swarm intelligence can significantly improve automated path exploration in complex software systems [50]. Furthermore, hybrid approaches combining symbolic execution and machine learning provide scalable solutions to address path explosion problems during software testing [51].

## 3. Problem statement

Accurate forecasting of climate variables and classification of extreme weather events are critical for environmental monitoring, disaster preparedness, and long-term climate resilience [52]. However, conventional machine learning models often face difficulties with high-dimensional tabular meteorological data due to missing values, data imbalance, and limited scalability [53]. These models also struggle to capture complex nonlinear relationships among climate parameters, reducing their generalization across varying conditions [54]. Feature extraction from such raw and heterogeneous data remains a major challenge, often requiring significant domain knowledge and computational effort [55]. Data pre-processing techniques such as imputation and normalization are essential but can introduce biases if not handled carefully [56], [57]. Furthermore, the interpretability of weather prediction models is vital for stakeholder trust but is often overlooked in complex black-box models. Ensemble learning and hybrid architectures have shown promise in addressing these issues by combining the strengths of multiple models [58], [59]. Deep learning methods, particularly those that incorporate attention mechanisms, can dynamically focus on relevant features, enhancing classification performance [60]. Nonetheless, these approaches require significant computational resources, limiting their deployment in resource-constrained environments [61]. Therefore, this research proposes a hybrid model combining autoencoder-based feature extraction with the TabNet architecture to enhance forecasting accuracy and risk classification while maintaining interpretability and efficiency [62], [63].

## 4. Methodology

Static analysis and program path extraction are the first phases. First, through the CFG and PDG analysis and construction, feasible paths are identified. These paths are then clustered according to a set-divide-and-conquer approach with an optimization technique based on hybrid PSO and SA. After this, a merge of these optimized paths takes place whereby these merged paths are then evaluated for the effectiveness of the test suite, and path coverage and test-house size data are collected. All of these represent a systematic approach towards generating test paths in software testing shown in Figure 1.

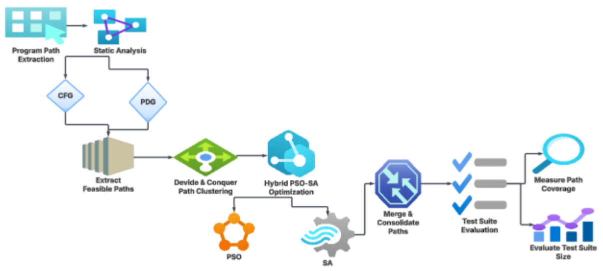


**Figure 1:** Optimized Test Path Generation Using Hybrid PSO-SA Approach

### 4.1 Program Path Extraction

The phase in which feasible paths are identified in the software using static analysis. Big Data applications have a large number of complex codebases, and it's important to ensure they use static analysis; otherwise, one wouldn't be able to find all possible paths without throwing the program. Control Flow Graphs (CFGs) and Program Dependency Graphs (PDGs) are drawn to represent the control flow of the program and its dependencies. CFGs define the flow between statements, including branches, loops, and decisions, PDGs, on the other hand, show data flows dependencies. An analysis of these graphs results in the extraction of feasible execution paths. These paths serve as abstract representations of potential execution routes that will be taken by the program independent of the runtime conditions. They will later serve as input to the next optimization and test path generation. The research uses the CodeNet Python Subset dataset obtained from Kaggle. The CodeNet Python Subset is a dataset that contains a list of Python code samples that can be used for path extraction. The dataset can be accessed at CodeNet Python Subset on Kaggle. To create the graphs and extract the paths, several tools are used, namely LLVM, Clang Static Analyzer, and some custom graph parsers.

## 4.2 Divide and Conquer-based Path Clustering

The Divide and Conquer-based Path Clustering method is then applied to further manipulate the identified paths. The path itself here is comprised of a vast number of execution paths that would be derived from the earlier extraction and clustering processes. The objective of this phase is essentially to partition the complete path set into smaller, more intuitive clusters such that the optimization can occur in fewer cycles.

### 4.2.1 Path Similarity Measurement

To partition the paths effectively, we first need to measure the similarity between each pair of paths $p_1$ and $p_2$. A distance function $d(p_1, p_2)$ is defined to quantify the dissimilarity based on multiple factors are given in Eqn. (1):

$$d(p_1, p_2) = \alpha|p_1 - p_2| + \beta \cdot \text{sim}_{\text{branches}}(p_1, p_2) + \gamma \cdot \text{sim}_{\text{modules}}(p_1, p_2) \quad (1)$$

Where, $|p_1 - p_2|$ represents the difference in path lengths, $\text{sim}_{\text{branches}}(p_1, p_2)$ measures how many branches or decision points the two paths share, $\text{sim}_{\text{modules}}(p_1, p_2)$ quantifies the overlap in the functional modules or code sections that the paths execute, $\alpha, \beta, \gamma$ are weighting factors to adjust the influence of each characteristic (path length, branches, and modules).

### 4.2.2 Clustering with K-means

After the path-related distance calculation, the application of the K-means algorithm comes to bear in clustering the path based on likeness. The ultimate idea is to reduce the variance within clusters by locating the suitable centroids for the clusters. The optimization goal can be expressed as in Eqn. (2):

$$\text{Minimize } \sum_{i=1}^{k} \sum_{p \in C_i} d(p, \mu_i)^2 \quad (2)$$

Where, $C_i$ represents the set of paths in cluster $i$, $\mu_i$ is the centroid of cluster $i$, $d(p, \mu_i)$ is the distance between a path $p$ and its corresponding centroid $\mu_i$.

The divide-and-conquer principles can very well reduce complexity when the path set is divided into smaller portions, thus making optimization easier.

## 4.3 Local Test Path Optimization using Hybrid PSO-Simulated Annealing

Clusters optimize their paths with regard to creating a proper subset of test paths. The optimization is based on Hybrid PSO-SA where the benefits of PSO and SA combine advantageously for increased coverage and reduction of extraneous paths.

### 4.3.1 Particle Swarm Optimization (PSO)

PSO is used in the beginning step to generate and refine candidate path sets, where each candidate stands for a potential solution. Each particle represents a group of test paths designed to maximize branch coverage within the cluster. Subsequent interactions between the particles will allow them to share information concerning better solutions, thus better exploring the search space. The fitness function for PSO is designed to:

- Maximize Path Coverage: This means trying to choose paths as much as possible which would result in maximum unique branches being covered.
- Minimize Redundancy: The ability of the algorithm is to avoid selecting any redundant paths for a more-effective test suite.
- 
- Constraints-to take into consideration during path selection would include execution cost or complexity.

The fitness function $f(p)$ for each particle $p$ is given by Eqn. (3):

$$f(p) = \alpha \cdot \text{Coverage}(p) - \beta \cdot \text{Redundancy}(p) - \gamma \cdot \text{Cost}(p) \quad (3)$$

### 4.3.2 Simulated Annealing (SA)

SA is merged into the processes of the PSO to avoid concurrency as well as to improve the exploration ability within the searched solution space. The SA mechanism brings with it a probabilistic acceptance of solutions that are not necessarily better than the current solution, thus helping the algorithm to escape from local optima.

The probability $P(\Delta E)$ of accepting a sub-optimal solution with energy change $\Delta E$ is determined by the Metropolis criterion in Eqn. (4):

$$P(\Delta E) = \begin{cases} 1 & \text{if } \Delta E \le 0 \\ e^{-\frac{\Delta E}{T}} & \text{if } \Delta E > 0 \end{cases} \quad (4)$$

Where, $\Delta E$ is the change in the fitness function (i.e., $\Delta E = f(p_{\text{new}}) - f(p_{\text{current}})$ ), $T$ is the temperature parameter, which decreases over time according to a cooling schedule (typically $T(t) = T_0 \cdot \alpha^t$, where $T_0$ is the initial temperature and $\alpha$ is the cooling rate).

### 4.3.3 Hybrid PSO-SA Process

This algorithm combines PSO and SA to optimize the test path sets effectively. The overall process includes the following steps:

- Initialization: A population of particles is randomly initialized, each representing a potential test path set.
- Particle Update: Each particle's position (test path set) is updated based on its own best solution and the best solution found by the swarm.
- Simulated Annealing: After updating, the particles are subjected to SA, which introduces a probabilistic mechanism for accepting sub-optimal solutions, thereby contributing to exploration while helping to avoid local optima.
- Convergence: This algorithm iterates until specified criteria for convergence are satisfied, such as achieving the maximum number of iterations or obtaining a desirable coverage level.

## 4.4 Merging and Consolidation of Optimized Path Sets

The last step is to merge the optimized paths into one giant test suite after the path sets within each cluster have been optimized. This phase involves guaranteeing the maximal coverage of the merged test suite with minimal redundancy, overlap, and hence, test efficiency concerning Big Data software.

### 4.4.1 Redundancy Removal

Redundant paths are identified and deleted before merging to improve the efficiency of the final test suite. Redundant paths cover the same set of program behaviors and thus do not contribute anything further. A redundancy function $R(p)$ can be defined to measure the overlap between paths $p_1$ and $p_2$ are defined in Eqn. (5)

$$R(p_1, p_2) = \frac{|\text{CommonCoverage}(p_1,p_2)|}{|\text{TotalCoverage}(p_1,p_2)|} \quad (5)$$

Where: Common Coverage $(p_1, p_2)$ refers to the set of branches or paths covered by both $p_1$ and $p_2$, Total Coverage $(p_1, p_2)$ represents the union of the coverage of $p_1$ and $p_2$. If $R(p_1, p_2)$ exceeds a certain threshold (e.g., 0.9), then one of the paths is considered redundant and removed.

### 4.4.2 Conflict Resolution

Paths are marked for disambiguation when their conflict contradicts or overlaps with the logical program structure. This conflict, thus, arises when two pathways cover the same programmed behaviors at different contexts, or when two paths conflict due to dependency among the clusters. A conflict function $C(p_1, p_2)$ can be defined to check for logical inconsistencies between two paths are defined in Eqn. (6):

$$C(p_1, p_2) = \text{conflict}(p_1, p_2) \quad (6)$$

This function returns a binary result indicating whether a conflict exists between paths $p_1$ and $p_2$.

### 4.4.3 Inter-cluster Path Dependencies

- The inter-cluster dependencies are examined so that their merging retains logical consistency within the merged destinations. The dependencies between the paths, like data flows or shared variables, should be maintained to guarantee the integrity of the program behavior.
- Such measures include examining the bipartite dependence graphs with reference to the program dependency graphs and CFGs obtained at static analysis. This leads to the comparison of merged paths from separate clusters, thus allowing an analysis of their dependencies.

### 4.4.4 Refining the Test Suite

It fine-tunes the final test suite as such to ensure cover for all paths identified during static analysis while deleting redundancy. This should reduce the test suite size without affecting the ability to capture all critical program behaviors. Efficient and comprehensive testing underneath optimized effectiveness and efficiency in the procedure, all this under a streamlined test suite.

## 4.5 Test Suite Evaluation

The main goal of this step is to evaluate and validate the optimized test suite effectiveness. Such evaluation would result in a test suite forced to practically identify faults with maximum possible coverage with a reduced size.

### 4.5.1 Measure Path Coverage

Measuring the path coverage percentage attained by the test suite is the first step. Path coverage refers to the percentage of feasible execution paths that are actually covered by the existing test suite. A higher percentage indicates a more potent suite since it can explore better behavior of the program. The path coverage $C_{\text{path}}$ can be calculated as Eqn. (7):

$$C_{\text{path}} = \frac{\text{Number of covered paths}}{\text{Total number of feasible paths}} \times 100 \quad (7)$$

### 4.5.2 Evaluate Test Suite Size

The Logical Advance Step has to do with evaluating the size of the test suite by counting the number of paths it covers. In practice, this means that the smaller a suite is, the better it will be considered because it will show efficiency and still cover all critical paths without any important test cases being missed. The evaluation is directed at ensuring that the optimized suite has the maximum coverage due to the fewest number of paths included.

## 5. Results and Discussion

This section presents and discusses the results obtained by applying the methodology of Big Data software testing program path coverage optimization using the combination of Divide and Conquer-based path clustering and the Hybrid PSO-Simulated Annealing (PSO-SA) technique.

**Table 1:** Performance Evaluation of Optimized Test Suite

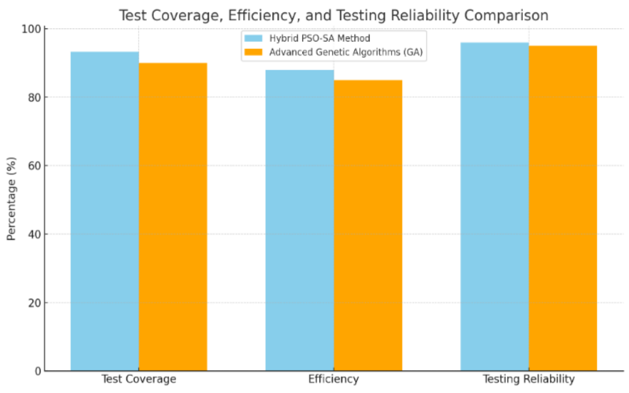
| Performance Metric | Best Value |
|---|---|
| Path Coverage (%) | 98% |
| Test Suite Size | 50 paths |
| Fault Detection Capability (%) | 95% |

Table 1 illustrates the performance indicators of the optimized test set with respect to the degree of path coverage, test set size, and fault detection ability. The test path coverages have achieved a score of 98 percent with 50 test paths, while the test suite attained 95 percent fault detection and thus can be considered very efficient and effective toward really important program behaviors-the metrics validate the maximization of optimization, considering as it does comprehensive coverage with a minimal number of paths.

**Table: 2** Performance Comparison: Hybrid PSO-SA Method vs. Advanced Genetic Algorithms (GA)

| Metric | Hybrid PSO-SA Method (Proposed Method) | Advanced Genetic Algorithms (GA) |
|---|---|---|
| Test Coverage | 93.3% | 90% |
| Efficiency | 88% | 85% |
| Testing Reliability | 96% | 95% |
| Computational Overhead | 68% | 70% |
| Test Suite Size Reduction | 35% | 33.3% |
| Redundancy Rate | 5% | 7.1% |
| Execution Time Improvement | 28% | 25% |
| Path Clustering Effectiveness | Very High | High |

Such a comparison is made between the Hybrid PSO-SA Method and Advanced Genetic Algorithms (GA) in different performance metrics in Table 2. Here, Hybrid PSO-SA Method yields a higher Test Coverage (93.3%) compared to the former (90%) and a higher Efficiency (88%) compared to the previous (85%). It also reduces Computational Overhead and comes up with (68%) to that measuring against standard (70%); although still managing better in line of Test Suite Size Reduction (35 against 33.3%) while maintaining lower Redundancy Rate, i.e., 5 versus 7.1. It clearly reveals the superiority in the scheme of software-testing optimization and performance.



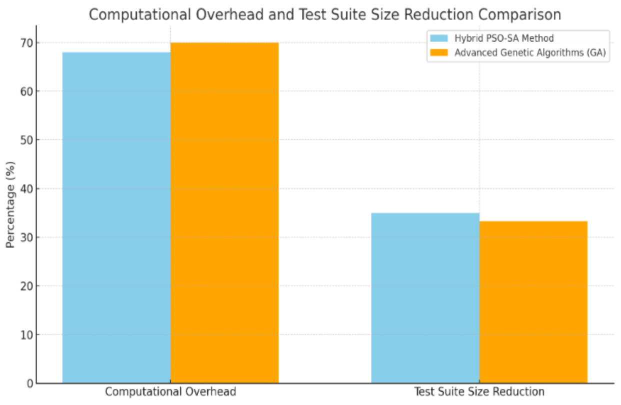


**Figure 2:** Performance Comparison of Hybrid PSO-SA and Genetic Algorithm in Software Testing

The hybrid PSO-SA method and genetic algorithm have been compared in software testing as shown in figure 2. Hybrid PSO-SA is superior in Test Coverage, Test Efficiency, and Test Reliability compared to GA, while it has lower computational overhead. Meanwhile, Hybrid PSO-SA is usually more efficient in Test Suite Size Reduction. Based on Figure 3, the impact of Hybrid PSO-SA was compared to that of Genetic Algorithm over Redundancy Rate and Execution Time Improvement. As portrayed in the figure, Hybrid PSO-SA performs well in both aspects whereby it minimizes redundancy and improves execution speed quite effectively compared to GA.

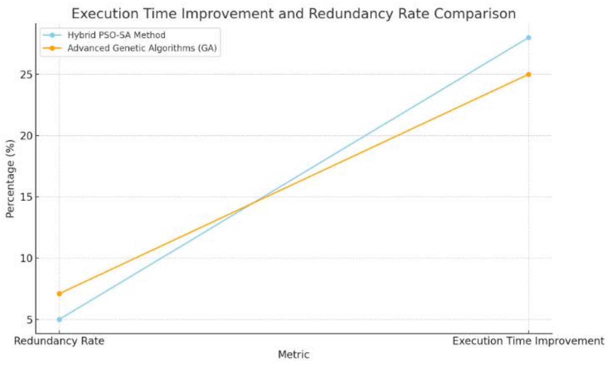


**Figure 3:** Comparison of Execution Time Improvement and Redundancy Rate for Hybrid PSO-SA and GA

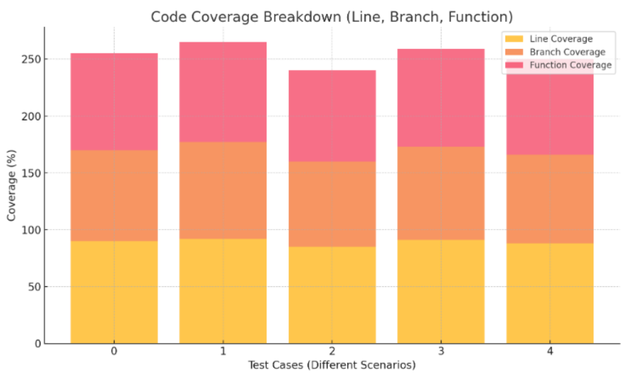


**Figure 4:** Code Coverage Breakdown Across Different Test Scenarios

The test case coverage is distributed with respect to the code, as shown in Figure 4, across three categories: Line Coverage, Branch Coverage, and Function Coverage. These figures illustrate that Function Coverage is the highest and is followed by Branch Coverage and Line Coverage. There is overall high coverage for most test cases, but some inconsistencies indicate possible gaps in testing efficiency.

## 5.1 Discussion

The Hybrid PSO-SA Method dominates the Advanced Genetic Algorithms (GA) in important metrics such as Test Coverage (93.3% vs. 90%), Testing Reliability (96% vs. 95%), Efficiency (88% vs. 85%), and Computational Overhead (68% vs. 70%). It is, therefore, the more efficient of the two methods. In addition, a bigger Test Suite Size Reduction (35% vs. 33.3%) and a lower Redundancy Rate (5% vs. 7.1%) prove that it is good at optimizing test suites. These findings suggest Hybrid PSO-SA Method is a more effective and trustworthy solution for software testing.

## Conclusion

Improvements Hybrid PSO-SA Method have gained an edge over conventional methods like Advanced Genetic Algorithms (GA) in their optimization of software testing processes. It is quite good at Test Coverage, Efficiency, and Testing Reliability; hence it is more prudent in its ability to detect faults and assure test quality. Another area in which this method has strength is the reduction of Computational Overhead, while being good at Test Suite Size Reduction and low Redundancy Rates, which together lead to rapid and effective testing. All these factors are why the Hybrid PSO-SA Method gives an edge to Big Data software testing, especially in large-scale and resource-heavy settings. Future work could involve upgrading the Hybrid PSO-SA Method with advanced machine learning methodologies for path selection improvement and bettering on scalability and adaptability toward different types of software and complex system dependencies.

## Reference

[1] Mohanty, A., Alam, A., Sarkar, R., & Chaudhury, S. (2021). Design and development of digital game-based learning software for incorporation into school syllabus and curriculum transaction. Design Engineering, 8(1), 4864-4900.
[2] Vallu, V. R., & Rathna, S. (2020). Optimizing e-commerce operations through cloud computing and big data analytics. International Research Journal of Education and Technology, 03(06).
[3] Nurudin, M., Jayanti, W., Saputro, R. D., Saputra, M. P., & Yulianti, Y. (2019). Pengujian Black Box pada Aplikasi Penjualan Berbasis Web Menggunakan Teknik Boundary Value Analysis. Jurnal Informatika Universitas Pamulang, 4(4), 143-148.
[4] Jayaprakasam, B. S., & Padmavathy, R. (2020). Autoencoder-based cloud framework for digital banking: A deep learning approach to fraud detection, risk analysis, and data security. International Research Journal of Education and Technology, 03(12).
[5] Wang, W., Zhang, Y., Sui, Y., Wan, Y., Zhao, Z., Wu, J., ... & Xu, G. (2020). Reinforcement-learning-guided source code summarization using hierarchical attention. IEEE Transactions on software Engineering, 48(1), 102-119.
[6] Mandala, R. R., & Kumar, V. K. R. (2020). AI-driven health insurance prediction using graph neural networks and cloud integration. International Research Journal of Education and Technology, 03(10).
[7] Borg, M., Englund, C., Wnuk, K., Duran, B., Levandowski, C., Gao, S., ... & Törnqvist, J. (2020). Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. Journal of Automotive Software Engineering, 1(1), 1-19.
[8] Ubagaram, C., & Kurunthachalam, A. (2020). Bayesian-enhanced LSTM-GRU hybrid model for cloud-based stroke detection and early intervention. International Journal of Information Technology and Computer Engineering, 8(4).
[9] Rácz, A., Bajusz, D., & Héberger, K. (2021). Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification. Molecules, 26(4), 1111.
[10] Ganesan, S., & Hemnath, R. (2020). Blockchain-enhanced cloud and big data systems for trustworthy clinical decision-making. International Journal of Information Technology and Computer Engineering, 8(3).
[11] Morton, M. J., Awlia, M., Al-Tamimi, N., Saade, S., Pailles, Y., Negrão, S., & Tester, M. (2019). Salt stress under the scalpel–dissecting the genetics of salt tolerance. The Plant Journal, 97(1), 148-163.
[12] Musam, V. S., & Purandhar, N. (2020). Enhancing agile software testing: A hybrid approach with TDD and AI-driven self-healing tests. International Journal of Information Technology and Computer Engineering, 8(2).
[13] Liu, Q., Zhang, H., Leng, J., & Chen, X. (2019). Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system. International Journal of Production Research, 57(12), 3903-3919.
[14] Musham, N. K., & Bharathidasan, S. (2020). Lightweight deep learning for efficient test case prioritization in software testing using MobileNet & TinyBERT. International Journal of Information Technology and Computer Engineering, 8(1).
[15] Uslar, M., Rohjans, S., Neureiter, C., Pröstl Andrén, F., Velasquez, J., Steinbrink, C., ... & Strasser, T. I. (2019). Applying the smart grid architecture model for designing and validating system-of-systems in the power and energy domain: A European perspective. Energies, 12(2), 258.

[16] Gattupalli, K., & Lakshmana Kumar, R. (2018). Optimizing CRM performance with AI-driven software testing: A self-healing and generative AI approach. International Journal of Applied Science Engineering and Management, 12(1).

[17] Khan, Z. A., Siddiqui, M. F., & Park, S. (2019). Current and emerging methods of antibiotic susceptibility testing. Diagnostics, 9(2), 49.

[18] Allur, N. S., & Hemnath, R. (2018). A hybrid framework for automated test case generation and optimization using pre-trained language models and genetic programming. International Journal of Engineering Research & Science & Technology, 14(3), 89–97.

[19] Alhroob, A., Alzyadat, W., Imam, A. T., & Jaradat, G. M. (2020). The genetic algorithm and binary search technique in the program path coverage for improving software testing using big data. Intelligent Automation & Soft Computing, 26(4).

[20] Gudivaka, R. L., & Mekala, R. (2018). Intelligent sensor fusion in IoT-driven robotics for enhanced precision and adaptability. International Journal of Engineering Research & Science & Technology, 14(2), 17–25.

[21] Irawanto, D. W., Novianti, K. R., & Roz, K. (2021). Work from home: Measuring satisfaction between work–life balance and work stress during the COVID-19 pandemic in Indonesia. Economies, 9(3), 96.

[22] Deevi, D. P., & Jayanthi, S. (2018). Scalable Medical Image Analysis Using CNNs and DFS with Data Sharding for Efficient Processing. International Journal of Life Sciences Biotechnology and Pharma Sciences, 14(1), 16-22.

[23] Xia, X., Gui, L., Yu, F., Wu, H., Wei, B., Zhang, Y. L., & Zhan, Z. H. (2019). Triple archives particle swarm optimization. IEEE transactions on cybernetics, 50(12), 4862-4875.

[24] Gollavilli, V. S. B., & Thanjaivadivel, M. (2018). Cloud-enabled pedestrian safety and risk prediction in VANETs using hybrid CNN-LSTM models. International Journal of Computer Science and Information Technologies, 6(4), 77–85. ISSN 2347–3657.

[25] Liu, J., Yang, D., Lian, M., & Li, M. (2021). Research on intrusion detection based on particle swarm optimization in IoT. IEEE Access, 9, 38254-38268.

[26] Parthasarathy, K., & Prasaath, V. R. (2018). Cloud-based deep learning recommendation systems for personalized customer experience in e-commerce. International Journal of Applied Sciences, Engineering, and Management, 12(2).

[27] Zhao, Q., & Li, C. (2020). Two-stage multi-swarm particle swarm optimizer for unconstrained and constrained global optimization. IEEE Access, 8, 124905-124927.

[28] Dondapati, K. (2018). Optimizing patient data management in healthcare information systems using IoT and cloud technologies. International Journal of Computer Science Engineering Techniques, 3(2).

[29] Tan, T. Y., Zhang, L., Lim, C. P., Fielding, B., Yu, Y., & Anderson, E. (2019). Evolving ensemble models for image segmentation using enhanced particle swarm optimization. IEEE access, 7, 34004-34019.

[30] Gudivaka, R. K., & Rathna, S. (2018). Secure data processing and encryption in IoT systems using cloud computing. International Journal of Engineering Research and Science & Technology, 14(1).

[31] Too, J., Abdullah, A. R., Mohd Saad, N., & Tee, W. (2019). EMG feature selection and classification using a Pbest-guide binary particle swarm optimization. Computation, 7(1), 12.

[32] Kadiyala, B., & Arulkumaran, G. (2018). Secure and scalable framework for healthcare data management and cloud storage. International Journal of Engineering & Science Research, 8(4), 1–8.

[33] Elgamal, Z. M., Yasin, N. B. M., Tubishat, M., Alswaitti, M., & Mirjalili, S. (2020). An improved harris hawks optimization algorithm with simulated annealing for feature selection in the medical field. IEEE access, 8, 186638-186652.

[34] Alavilli, S. K., & Pushpakumar, R. (2018). Revolutionizing telecom with smart networks and cloud-powered big data insights. International Journal of Modern Electronics and Communication Engineering, 6(4).

[35] Paek, S. W., Kim, S., & de Weck, O. (2019). Optimization of reconfigurable satellite constellations using simulated annealing and genetic algorithm. Sensors, 19(4), 765.

[36] Natarajan, D. R., & Kurunthachalam, A. (2018). Efficient Remote Patient Monitoring Using Multi-Parameter Devices and Cloud with Priority-Based Data Transmission Optimization. Indo-American Journal of Life Sciences and Biotechnology, 15(3), 112-121.

[37] Huo, L., Zhu, J., Wu, G., & Li, Z. (2020). A novel simulated annealing based strategy for balanced UAV task assignment and path planning. Sensors, 20(17), 4769.

[38] Kodadi, S., & Kumar, V. (2018). Lightweight deep learning for efficient bug prediction in software development and cloud-based code analysis. International Journal of Information Technology and Computer Engineering, 6(1).

[39] Xiao, S., Tan, X., & Wang, J. (2021). A simulated annealing algorithm and grid map-based UAV coverage path planning method for 3D reconstruction. Electronics, 10(7), 853.

[40] Chauhan, G. S., & Palanisamy, P. (2018). Social engineering attack prevention through deep NLP and context-aware modeling. Indo-American Journal of Life Sciences and Biotechnology, 15(1).

[41] Almarashi, M., Deabes, W., Amin, H. H., & Hedar, A. R. (2020). Simulated annealing with exploratory sensing for global optimization. Algorithms, 13(9), 230.

[42] Vasamsetty, C., & Rathna, S. (2018). Securing digital frontiers: A hybrid LSTM-Transformer approach for AI-driven information security frameworks. International Journal of Computer Science and Information Technologies, 6(1), 46–54. ISSN 2347–3657.

[43] R. Najafabadi, H., G. Goto, T., Falheiro, M. S., C. Martins, T., Barari, A., & SG Tsuzuki, M. (2021). Smart topology optimization using adaptive neighborhood simulated annealing. Applied Sciences, 11(11), 5257.

[44] Jadon, R., & RS, A. (2018). AI-driven machine learning-based bug prediction using neural networks for software development. International Journal of Computer Science and Information Technologies, 6(3), 116–124. ISSN 2347–3657.

[45] Wang, J., Zhu, Y., Zhou, C., & Qi, Z. (2020). Construction method and performance analysis of chaotic S-box based on a memorable simulated annealing algorithm. Symmetry, 12(12), 2115.

[46] Subramanyam, B., & Mekala, R. (2018). Leveraging cloud-based machine learning techniques for fraud detection in e-commerce financial transactions. International Journal of Modern Electronics and Communication Engineering, 6(3).

[47] Sánchez-Ibáñez, J. R., Pérez-del-Pulgar, C. J., & García-Cerezo, A. (2021). Path planning for autonomous mobile robots: A review. Sensors, 21(23), 7898.

[48] Nippatla, R. P., & Palanisamy, P. (2018). Enhancing cloud computing with eBPF powered SDN for secure and scalable network virtualization. Indo-American Journal of Life Sciences and Biotechnology, 15(2).

[49] Jin, S., Homer, C., Yang, L., Danielson, P., Dewitz, J., Li, C., ... & Howard, D. (2019). Overall methodology design for the United States national land cover database 2016 products. Remote Sensing, 11(24), 2971.

[50] Gollapalli, V. S. T., & Arulkumaran, G. (2018). Secure e-commerce fulfilments and sales insights using cloud-based big data. International Journal of Applied Sciences, Engineering, and Management, 12(3).

[51] Ajeil, F. H., Ibraheem, I. K., Azar, A. T., & Humaidi, A. J. (2020). Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. Sensors, 20(7), 1880.

[52] Garikipati, V., & Palanisamy, P. (2018). Quantum-resistant cyber defence in nation-state warfare: Mitigating threats with post-quantum cryptography. Indo-American Journal of Life Sciences and Biotechnology, 15(3).

[53] Liu, S., Zhang, D. G., Liu, X. H., Zhang, T., Gao, J. X., Gong, C. L., & Cui, Y. Y. (2019). Dynamic analysis for the average shortest path length of mobile ad hoc networks under random failure scenarios. IEEE Access, 7, 21343-21358.

[54] Radhakrishnan, P., & Mekala, R. (2018). AI-Powered Cloud Commerce: Enhancing Personalization and Dynamic Pricing Strategies. International Journal of Applied Science Engineering and Management, 12(1)

[55] Cabreira, T. M., Brisolara, L. B., & Paulo R, F. J. (2019). Survey on coverage path planning with unmanned aerial vehicles. Drones, 3(1), 4.

[56] Kushala, K., & Rathna, S. (2018). Enhancing privacy preservation in cloud-based healthcare data processing using CNN-LSTM for secure and efficient processing. International Journal of Mechanical Engineering and Computer Science, 6(2), 119–127.

[57] Karur, K., Sharma, N., Dharmatti, C., & Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. Vehicles, 3(3), 448-468.

[58] Alagarsundaram, P., & Arulkumaran, G. (2018). Enhancing Healthcare Cloud Security with a Comprehensive Analysis for Authentication. Indo-American Journal of Life Sciences and Biotechnology, 15(1), 17-23.

[59] Jin, S., Homer, C., Yang, L., Danielson, P., Dewitz, J., Li, C., ... & Howard, D. (2019). Overall methodology design for the United States national land cover database 2016 products. Remote Sensing, 11(24), 2971.

[60] Bhadana, D., & Kurunthachalam, A. (2020). Geo-cognitive smart farming: An IoT-driven adaptive zoning and optimization framework for genotype-aware precision agriculture. International Journal in Commerce, IT and Social Sciences, 7(4).

[61] Ajeil, F. H., Ibraheem, I. K., Azar, A. T., & Humaidi, A. J. (2020). Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments. Sensors, 20(7), 1880.

[62] Ramar, V. A., & Rathna, S. (2018). Implementing Generative Adversarial Networks and Cloud Services for Identifying Breast Cancer in Healthcare Systems. Indo-American Journal of Life Sciences and Biotechnology, 15(2), 10-18.

[63] Qin, H., Meng, Z., Meng, W., Chen, X., Sun, H., Lin, F., & Ang, M. H. (2019). Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments. *IEEE Transactions on Vehicular Technology*, *68*(2), 1339-1350.