JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 6 No. 3 (2025) 135-153
https://publisher.uthm.edu.my/ojs/index.php/jscdm

The Hybrid Local Maximum Distance Algorithm with Dissimilarity-Based Test Case Prioritization for Software Product Line Testing

Siti Hawa Mohamed Shareef[1], Rabatul Aduni Sulaiman[1]*, Abd Samad Hasan Basari[1], Muhammad Arif Shah[2]

[1] *Faculty of Computer Science and Information Technology (FSKTM), Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, Batu Pahat, Johor 86400, MALAYSIA*
[2] *ITCS, City University of Science and Information Technology, Peshawar, PAKISTAN*

*Corresponding Author: rabatul@uthm.edu.my
DOI: https://doi.org/10.30880/jscdm.2025.06.03.010

Article Info

Received: 29 July 2025
Accepted: 10 December 2025
Available online: 30 December 2025

Keywords

Test case prioritization, software product line testing, hybrid string distance, dissimilarity measure, regression testing, ELMDA

Abstract

Software Product Line (SPL) testing presents significant challenges due to configuration variability. Testers must ensure the quality of both common and variant-specific behaviours across product variants. Test Case Prioritization (TCP) is a well-established strategy for improving regression testing efficiency by enabling early fault detection. This paper proposes a hybrid prioritization approach that integrates a dissimilarity-based method with the Enhanced Local Maximum Distance Algorithm (ELMDA). The proposed approach introduces three hybrid string distance techniques, New Enhanced Hybrid Technique (NEHT1, NEHT2, and NEHT3), combining Jaro-Winkler and Manhattan distances to quantify test case diversity more effectively. These dissimilarity scores guide the selection of structurally and behaviourally distinct test cases, enhancing both fault detection and execution efficiency. Empirical evaluations were conducted using two SPL case studies: the Global Positioning System (GPS) and the iRobot Roomba, each tested against five mutant versions. Results show that NEHT1 consistently outperforms all other variants, achieving the best trade-off between Average Percentage of Faults Detected (APFD) and execution time, with improvements of approximately 4.9-11.8% in APFD and a 38-54% reduction in execution time over the best baseline ($p < 0.05$). The findings also demonstrate that the proposed approach significantly outperforms existing techniques in terms of APFD and execution time. Notably, NEHT1 achieves the highest APFD and the lowest execution time across both experiments, confirming its effectiveness as the top-performing variant. Statistical analysis further validates the significance of these improvements. Overall, the results establish ELMDA with NEHT1 as a promising solution for effective and scalable SPL regression testing.

1. Introduction

Software Product Line (SPL) development enables organizations to generate a family of related software products by systematically reusing core assets across different configurations [1]. While this promotes efficiency and reduces time-to-market [2][3], it introduces complexity in software testing due to the high variability among product variants. To ensure both shared and unique features behave as expected, effective testing approaches, particularly within the Software Product Line Testing (SPLT) paradigm, are essential.

SPLT differs from traditional single-system testing as it requires the simultaneous validation of multiple product variants. This added complexity makes early fault detection and cost-effective execution critical challenges [4]. Test Case Prioritization (TCP) has emerged as an important strategy in this context, as it reorders test cases to detect faults earlier, thereby improving test efficiency and reducing resource consumption.

However, many existing TCP techniques rely on historical fault data or static similarity measures. This fails to account for the evolving nature and variability of SPL configurations [5][6]. These limitations can lead to suboptimal prioritization and inefficient regression testing. Dissimilarity-based approaches have shown potential to address this issue by selecting more diverse and effective test cases [7]. However, there are still issues in integrating both fault detection capability and execution cost in a balanced and scalable manner [8]. Moreover, current dissimilarity-based methods often overlook the structural complexity of SPL models, leading to inconsistent prioritization outcomes. The lack of standardized strategies for measuring and applying dissimilarity also hinders broader applicability across diverse SPL domains. Consequently, SPL variability often results in redundant test executions, inefficient execution cost, and delayed fault revelation, highlighting the need for a cost-aware, diversity-driven TCP approach.

To address this gap, this study introduces the Enhanced Local Maximum Distance Algorithm (ELMDA), a dissimilarity-based prioritization approach designed specifically for SPL regression testing. ELMDA enhances the traditional Local Maximum Distance approach by incorporating execution efficiency into the prioritization process. The algorithm is evaluated using two real-world SPL case studies, which are the Global Positioning System (GPS) and the i-Robot Roomba. Then it will be tested against five mutant versions. For these experiments, the GPS model contains eight test cases, while the i-Robot Roomba model comprises twenty test cases. The results demonstrate that ELMDA outperforms existing algorithms, such as Last Minimal for Local Maximal Distance (LM-LMD) [7] and the Enhanced All-yes Config Algorithm (EA) [9] in terms of Average Percentage of Faults Detected (APFD) and execution time [10][11].

This research aims to enhance dissimilarity-based TCP for SPLT by integrating diversity-aware and cost-effective measures into a unified prioritization strategy. SPL variability often leads to redundant test executions, increased testing cost, and delayed fault revelation when traditional TCP techniques fail to capture behavioural diversity effectively. This limitation underscores the need for a prioritization strategy that can simultaneously address diversity and execution efficiency.

To achieve this, the following objectives are defined:

- To develop an ELMDA that incorporates hybrid string distance techniques (NEHT1-NEHT3) to improve fault detection capability in SPL regression testing.
- To empirically evaluate the effectiveness and efficiency of the proposed ELMDA using two SPL case studies (GPS and iRobot Roomba) and validate its performance through APFD and execution time metrics supported by statistical analysis.

This work advances the field of software testing by:

- Introducing a hybrid prioritization algorithm (ELMDA) that integrates dissimilarity-based techniques to optimize test-case ordering for APFD and execution time using two SPL case studies.
- Providing statistical validation using the Kruskal-Wallis test and Holm-Bonferroni correction, confirming the significance of the observed improvements in both APFD and execution efficiency.

The remainder of this paper is organized as follows: Section 2 presents related work on SPLT and TCP; Section 3 describes the background on SPL variability and regression testing; Section 4 introduces the proposed ELMDA algorithm; Section 5 discusses the experimental design and results; Section 6 outlines threats to validity; and Section 7 concludes the paper and suggests future research directions.

2. Related Work

SPL promotes large-scale reuse by developing multiple product variants from a shared set of core assets, such as architecture, components, and features [1]. This strategy improves development efficiency and product consistency while reducing cost and time-to-market [2][3]. However, SPL introduces testing challenges due to configuration diversity and feature variability. Each variant must be thoroughly tested to ensure correctness, increasing demand for scalable and efficient testing approaches.

To manage this complexity, SPLT has been established as a framework for validating product variants derived from feature models. SPLT must ensure both common and variant-specific behaviours are adequately tested. Model-Based Testing (MBT) is often employed in SPLT to automate test case generation from behavioural models, enhancing testing efficiency and reducing redundancy [7]. However, MBT alone does not address the sequencing or selection of test cases during execution, which limits its effectiveness in regression testing scenarios.

Regression testing plays a critical role in ensuring that modifications to a software system do not introduce new faults into existing functionality[12]. As systems evolve, regression testing helps verify the stability and correctness of both common and variant-specific behaviours, especially in SPL environments.

TCP aims to reorder test cases so that those most likely to detect faults are executed earlier. This approach is especially valuable in regression testing, where the goal is to maximize fault detection while minimizing execution effort [10]. Traditional TCP techniques frequently rely on similarity metrics [13], code coverage data [14], or past test outcomes [15]. In SPL contexts, where configurations evolve and code reuse is common, these strategies may produce overlapping or redundant test cases that fail to maximize diversity [5] [6].

To address these shortcomings, dissimilarity-based TCP techniques have gained attention. These approaches prioritize test cases that differ structurally or behaviourally, thereby exploring a wider range of potential faults [11]. However, a key limitation in prior work is the lack of attention to execution cost. While some studies emphasize early fault detection, only a few consider execution time as a prioritization metric, despite its critical importance in large-scale SPLT.

The Local Maximum Distance Algorithm (LMDA) introduced a promising direction by ranking test cases based on their distance from previously selected cases, improving diversity. Yet, it lacks optimization for cost and may introduce inefficiencies when applied to evolving product lines [7]. Similarly, the Enhanced All-yes Config Algorithm focuses on fault-based prioritization but is limited in adaptability when applied to mutant versions [9].

The recent advancements in TCP continue to emphasize adaptive learning, diversity-awareness, and cost-effectiveness to improve fault detection in evolving systems. For instance, Test Prioritization Visualization (TPVis) introduced a visual analytics platform to explore and analyse prioritization performance, demonstrating the importance of interactive tools in regression testing evaluation [16]. For instance, Durelli *et al.* [17] proposed a model-based, diversity-driven, learn-to-rank framework that integrates behavioural dissimilarity into a learning model, which is particularly suitable for high-variability domains such as SPL. Likewise, Lemos *et al.* [13] developed a cluster-based adaptive prioritization approach that dynamically adapts to fault clusters, thereby improving early fault detection rates. In the domain of test generation, Honfi *et al.* [18] introduced EvoDomain, a search-based technique designed to generate domain-aware test suites that enhance structural diversity and improve coverage in logic-based systems. Furthermore, Wang *et al.* [19] proposed a parallel evolutionary framework that optimizes test generation through subpopulation strategies, resulting in scalable and diverse test cases for web applications. Collectively, these studies support the integration of diversity and performance efficiency in TCP, reinforcing the relevance and novelty of the proposed work in addressing regression testing challenges within SPL environments.

Table 1 summarizes prior TCP approaches in terms of SPL relevance, diversity consideration, and execution cost awareness. The comparison shows that most SPL-focused techniques emphasize diversity but do not consider execution time, while methods that address cost or time are generally not tailored for SPL testing. This highlights a clear gap in existing work, reinforcing the need for a unified TCP approach that integrates both dissimilarity and execution cost for SPL environments.

Table 1 *Comparative mapping of existing TCP techniques*

Approach	SPL Context	Diversity-Based	Considers Execution Cost / Time	Data / Code Used
[7]	Yes	Yes	No (focus APFD only)	Statechart MBT test suites
[9]	Yes	Yes	No (execution time not considered)	FM-based test suites
[16]	No	Yes	No (APFD only)	EFSM models
[17]	No	Partial	Yes (execution time overhead analysed)	Java programs, clusters
[13]	No	Yes	No (optimises search, but not TCP execution cost)	Boundary input domains
[18]	No	No	Yes (execution duration visualised but not used as metric)	Large test logs
[19]	No	Yes	Yes (log-execution time observed)	Execution logs
[10]	Yes	Yes	Yes (CI test budget, cost-aware)	CI logs, variant test sets
[11]	No	Yes	Yes (reduces evolution/execution time)	Web applications

To fill this gap, this study proposes a hybrid ELMDA that incorporates dissimilarity values and execution time into a unified prioritization scheme. The algorithm is experimentally validated on two SPL case studies, demonstrating significant improvements in both APFD and execution time. The proposed hybrid approach can serve as a cost-effective, diversity-aware alternative to TCP in SPL.

3. The Proposed Approach

This section presents the proposed hybrid prioritization approach. Here, the approach integrates dissimilarity-based test case selection with hybrid string distance techniques. The framework is centered around the ELMDA, which incrementally constructs a prioritized test suite by maximizing behavioral diversity. To compute dissimilarity more effectively, ELMDA incorporates three enhanced hybrid string distance formulations, NEHT1, NEHT2, and NEHT3, that combine both structural and character-level variations between test cases.

3.1 Overview of the ELMDA Algorithm

The ELMDA builds upon the traditional LMDA by integrating a dissimilarity-based selection mechanism that prioritizes test cases that exhibit the highest behavioral diversity. The core principle of ELMDA is to incrementally build a prioritized test suite by repeatedly selecting the test case that is most dissimilar to those already selected. This approach helps avoid redundancy and promotes coverage of distinct behaviors, which is especially beneficial in SPL regression testing where variability among product configurations must be validated efficiently.

The enhancements introduced in ELMDA primarily concern the dissimilarity calculation and the selection mechanism. ELMDA incorporates hybrid string distance techniques to quantify dissimilarity more precisely and modifies the candidate selection strategy to better reflect the diversity of test cases across iterations. These enhancements are introduced at key stages of the algorithm to increase fault-detection effectiveness and reduce redundant test executions.

3.2 Hybrid String Distance Techniques: NEHT1, NEHT2, NEHT3

In order to represent feature sequences in the approach, three New Enhanced Hybrid Techniques (NEHT1, NEHT2, and NEHT3) are used [8]. These formulations combine Jaro–Winkler similarity, which captures character-level proximity, with Manhattan distance, which emphasizes structural or positional differences in test case elements.

Each hybrid technique is normalized to produce a consistent dissimilarity scale ranging from 0 (identical) to 1 (completely dissimilar). These scores are used by ELMDA to select test cases that maximize behavioural diversity iteratively.

3.2.1 Technique 1 (NEHT1)

NEHT1 is a hybrid string distance technique that combines Manhattan distance and the Jaro-Winkler metric to improve dissimilarity measurement. It decomposes the Manhattan distance into subcomponents and refines Jaro-Winkler by focusing on prefix similarity. This enables NEHT1 to capture both overall differences and detailed similarities between test cases. The formulation, shown in Equation 1, is appropriate for SPLT contexts where both structural and prefix-based distinctions matter.

$$\frac{1}{3}\left(\frac{\frac{T_1+T_2}{2}}{T_1}+\frac{\frac{T_1+T_2}{2}}{T_2}+\frac{m(n+m)-(|T_1-T_2|+|T_1-T_2|)}{m(n+m)}\right) \tag{1}$$

where:

m	is the maximum number of matching characters between test cases 1 and 2.
n	number of deselected features in both test cases.
T_1, T_2	is the length of test cases 1 and 2

The variable m denotes the number of matching characters between two test cases, while n refers to the number of features that are deselected, distinguishing them. T_1 and T_2 represent the lengths of test case 1 and test case 2, respectively. These values collectively measure both similarity and structural differences for effective dissimilarity-based prioritization.

3.2.2 Technique 2 (NEHT2)

NEHT2 builds upon NEHT1 by further separating and analyzing the components of similarity. It divides the Manhattan distance into spatial sub-values and splits the Jaro–Winkler metric into its Jaro and Winkler components to assess overall character similarity and prefix alignment separately. This refined structure enables more precise evaluation of test case similarity. The full formulation is shown in Equation 2.

$$\frac{1}{3}\left(\frac{m}{T_1}+\frac{m}{T_2}+\frac{(m+n)-\left(\frac{|T_1-T_2|}{|T_1-T_2|}\right)}{(m+n)}\right)+\left(\frac{|T_1-T_2|}{|T_1+T_2|}(1-d_j)\right) \tag{2}$$

where:

m	is the count of the maximum number of matching characters between test cases 1 and 2.
n	number of deselected features in both test cases.
T_1, T_2	is the length of test cases 1 and 2.
d_j	is the Jaro distance

The variable m is the number of matching characters between test case 1 and test case 2, while n indicates the number of deselected features in both test cases. T_1 and T_2 represent the lengths of test cases 1 and 2, respectively. d_j denotes the Jaro distance, which quantifies the similarity between two strings based on character matches and transpositions.

3.2.3 Technique 3 (NEHT3)

NEHT3 enhances the hybrid similarity metric by expanding on the designs of NEHT1 and NEHT2. It further splits the Manhattan distance to highlight spatial differences and separates the Jaro-Winkler metric into its Jaro and Winkler components for detailed analysis. This structure enables NEHT3 to capture both broad structural variation and fine-grained character similarity. The complete formulation is shown in Equation 3.

$$\frac{1}{3}\left(\frac{m}{T_1}+\frac{m}{T_2}+\frac{(m+n)-\left(\frac{|T_1-T_2|}{|T_1-T_2|}\right)}{(m+n)}\right)+\left(\frac{|T_1-T_2|}{|T_1+T_2|}(1-d_j)\right) \tag{3}$$

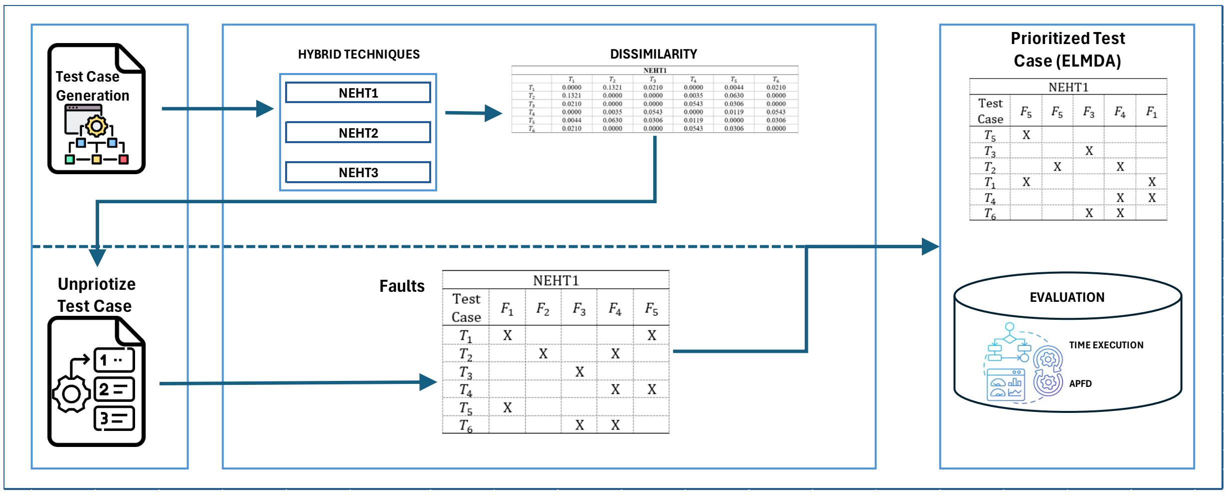
where:

m	is the count of the maximum number of matching characters between test cases 1 and 2.
n	number of deselected features in both test cases.
T_1, T_2	is the length of test cases 1 and 2.
d_j	is the Jaro distance

The variable m represents the number of matching characters between test case 1 and test case 2, while n denotes the number of deselected features in both test cases. T_1 and T_2 indicate the lengths of test cases 1 and 2, respectively. The term d_j refers to the Jaro distance, which measures the similarity between two strings based on character matches and their positions.

3.3 Prioritization Workflow

Fig. 1 illustrates the overall workflow of the TCP process, integrating the proposed hybrid string distance techniques with the ELMDA. The process begins with Test Case Generation, in which unprioritized test cases are generated from feature models or system specifications. These test cases are then processed through the Hybrid Techniques, namely NEHT1, NEHT2, and NEHT3. Each technique combines string distance metrics, such as Jaro-Winkler and Manhattan, to generate a dissimilarity matrix that quantifies variation between test cases. In this step, each NEHT technique performs a structured merging process in which the individual distance values from both metrics are combined using their respective hybrid formulas to produce a single unified dissimilarity score for every pair of test cases. This merging operation is essential for translating multiple behavioral signals into a single representative hybrid distance value. The example in the diagram highlights the NEHT1 dissimilarity values used to guide the prioritization process.



NEHT1						
	T_1	T_2	T_3	T_4	T_5	T_6
T_1	0.0000	0.1321	0.0210	0.0000	0.0044	0.0210
T_2	0.1321	0.0000	0.0000	0.0035	0.0630	0.0000
T_3	0.0210	0.0000	0.0000	0.0543	0.0306	0.0000
T_4	0.0000	0.0035	0.0543	0.0000	0.0119	0.0543
T_5	0.0044	0.0630	0.0306	0.0119	0.0000	0.0306
T_6	0.0210	0.0000	0.0000	0.0543	0.0306	0.0000

NEHT1					
Test Case	F_1	F_2	F_3	F_4	F_5
T_1	X				X
T_2		X		X	
T_3			X		
T_4				X	X
T_5	X				
T_6			X	X	

NEHT1					
Test Case	F_5	F_5	F_3	F_4	F_1
T_5	X				
T_3			X		
T_2		X		X	
T_1	X				X
T_4				X	X
T_6			X	X	

Fig. 1 *ELMDA process flow*

Next, the fault detection table is constructed to represent which faults (F_1 to F_5) are detected by each test case (T_1 to T_6) in their unprioritized form. This mapping is essential for the subsequent calculation of the APFD metric. Using the dissimilarity matrix as input, the test cases are reordered through the Prioritized Test Case process using ELMDA. The ELMDA technique selects test cases based on their relative dissimilarity, ensuring greater diversity and a higher likelihood of early fault detection. The prioritized table, as shown for NEHT1, arranges test cases according to their ability to detect faults from F_5 to F_1, aiming to maximise early fault detection.

Finally, the effectiveness of the prioritization is assessed in the evaluation phase using two key metrics: Execution Time, which captures the efficiency of test-case execution, and APFD, which measures how quickly faults are detected across the ordered test suite. This end-to-end workflow demonstrates how hybrid string distance techniques and the ELMDA algorithm collaborate to enhance TCP in SPLT, improving both execution cost and fault-detection effectiveness.

3.4 Algorithm Description

To address the limitations of the original LMDA, this study introduces the ELMDA, designed to improve TCP in SPLT. ELMDA enhances prioritization by incorporating dissimilarity metrics derived from three hybrid string distance techniques, NEHT1, NEHT2, and NEHT3, each of which combines elements of the Jaro-Winkler and Manhattan distance measures. These hybrid techniques aim to provide a more precise and adaptable assessment of test case differences, capturing both character-level similarity and spatial variation. The goal of ELMDA is to improve early fault detection, reduce redundant test executions, and accommodate the variability inherent in SPL environments.

Algorithm: ELMDA

```
1.  Input: S = {C1, C2, ..., Cm}  // Set of test cases
2.  Output: CS                             // List of test cases in prioritized order

Initialize prioritized list
3.  CS ← []

Main loop to process all test cases
4.  while #S > 0 do
5.      if CS is empty then
6.         Ci ← Select Ci ∈ S such that
7.                sum(coverage_weight(Ci, Cj) for all Cj ∈ S) is maximized
8.       else

Subsequent selections: maximize marginal contribution to CS
9.         Ci ← Select Ci ∈ S such that
10.               sum(coverage_weight(Ci, Cj) for all Cj ∈ CS U {Ci}) is maximized

Select case contributing most to CS
11.     end if
12.     CS.add(Ci)
13.     S ← S \ {Ci}
14. end while
15. return CS
```

The structure of ELMDA is presented in the pseudocode (Algorithm: ELMDA). The algorithm begins by accepting an initial set of unprioritized test cases, denoted as S, and initializing an empty list, CS, which will store the prioritized test cases. The prioritization proceeds iteratively until all test cases in S have been processed and moved to CS. At each iteration, ELMDA selects the most suitable test case based on hybrid dissimilarity values calculated from the NEHT1, NEHT2, or NEHT3 metrics.

To account for behavioural differences between test cases during prioritization, ELMDA uses a coverage weighting function defined in the equation below. The complete formulation is shown in Equation 4. This weighting ensures that test cases with greater behavioural variation contribute higher dissimilarity scores within the ELMDA selection logic. The resulting values are incorporated directly into the dissimilarity computation used in Lines 7 and 10 of the algorithms.

$$coverage\_weight(C_i, C_j) = \frac{|cov(C_i)\ \Delta\ cov(C_j)|}{|cov(C_i)\ \cup\ cov(C_j)|} \tag{4}$$

where:

$cov(C_i)$	set of transitions, states, or feature interactions covered by test case C_i
$cov(C_j)$	set of transitions, states, or feature interactions covered by test case C_j

Range: $0 \leq coverage\_weight(C_i, C_j) \leq 1$

Interpretation:
0 → identical behavioural coverage
1 → completely different behavioural coverage

In the initial stage, when CS is empty, the algorithm selects the test case from S that exhibits the highest total dissimilarity with respect to all other test cases in the set. This strategy ensures that the starting point of the prioritized sequence represents a test case with the most distinctive feature interactions and behavioural characteristics. In subsequent iterations, the algorithm selects the test case from S that contributes the highest marginal dissimilarity relative to those already included in CS This is computed by summing the pairwise dissimilarity values between the candidate test case and all members of the prioritized list. The selection process continues until all test cases have been transferred from S to CS, resulting in a test suite structured to enhance behavioural diversity and support early fault detection.

The dissimilarity values guiding the selection process are derived from the NEHT1, NEHT2, and NEHT3 hybrid distance techniques. Each value in the dissimilarity matrix represents the degree of difference between pairs of test cases, capturing both feature-level interaction and behavioral variation. These matrices are directly used in the algorithm through the $coverage\_weight(C_i, C_j)$ function, which evaluates the prioritization impact of each test case relative to others.

The rationale for maximizing dissimilarity in this context is grounded in the need to improve the APFD, a critical metric in regression testing. In SPL environments, many test cases share core functionalities, leading to overlapping behaviours. Prioritizing highly similar test cases early in the sequence may delay the discovery of faults. In contrast, selecting structurally and behaviourally distinct test cases first increases the likelihood of exposing faults, particularly those related to variant-specific features, at an earlier stage. This approach not only enhances fault detection but also improves testing efficiency by reducing redundancy and promoting greater coverage diversity.

3.4.1 Improvement of Local Maximum Distance Algorithm

The ELMDA improves upon the original LMDA by integrating hybrid dissimilarity-based calculations into the selection logic. This enhancement enables the algorithm to prioritize test cases that are both structurally and behaviourally diverse, while also considering execution cost efficiency, which is essential for SPLT.

The improvement is reflected directly in Line 7 and Line 10 of the algorithm, where $coverage\_weight(C_i, C_j)$ function is used to guide the selection of the next test case. In ELMDA, this function no longer relies solely on structural distance; instead, it uses hybrid dissimilarity scores calculated using the NEHT1, NEHT2, or NEHT3 formulas.

Each hybrid technique combines Jaro-Winkler similarity (capturing lexical similarity) and Manhattan distance (capturing structural difference) to create a more robust and context-aware dissimilarity measure. These scores are precomputed in a dissimilarity matrix, which is referenced during test case selection.

Integration into ELMDA Pseudocode:

i. Line 6-7 (First test case selection):

- The algorithm selects the test case $CS \in S$ that is most dissimilar from all others in the remaining set S. The dissimilarity is computed using the selected NEHT formula. This ensures that the first test case is highly diverse and maximizes initial test space coverage. The complete formulation is shown in Equation 5.

$$C_i \leftarrow \underset{C_i \in S}{arg\,max} \sum_{C_i \in S} D\left(C_i, C_j\right) \tag{5}$$

- Where $D\left(C_i,\ C_j\right)$ is the hybrid dissimilarity score from NEHT1, NEHT2, or NEHT3.

ii. Line 9-10 (Subsequent selections):
   - For each subsequent iteration, the algorithm selects the test case $CS \in S$ that provides the highest cumulative dissimilarity relative to the already prioritized test cases CS . This promotes behavioural diversity in the early stages of execution. The complete formulation is shown in Equation 6.

$$C_i \leftarrow \underset{C_i \in S}{arg\,max} \sum_{CS \cup \{C_i\}} D\left(C_i, C_j\right) \tag{6}$$

iii. Line 12-13 (Add and remove):
   - Once the test case with the highest dissimilarity score is identified, it is added to the prioritized list CS and removed from S, continuing until all test cases are prioritized.

In the ELMDA algorithm, the integration of NEHT1, NEHT2, and NEHT3 occurs during the selection phases at Lines 7 and 10, where the algorithm determines which test case to prioritize next based on dissimilarity. Specifically, the generic $coverage\_weight(C_i, C_j)$ function in the pseudocode is replaced by a hybrid dissimilarity score calculated using one of the proposed NEHT formulas. NEHT1 combines length-normalized ratios and structural differences to evaluate diversity between test cases, while NEHT2 extends this by incorporating a dissimilarity correction factor based on similarity d_j NEHT3 builds upon NEHT2 by further emphasizing behavioural variation. These dissimilarity scores are computed for each pair of test cases and used to form a dissimilarity matrix. During the initial selection (Line 7), the test case with the highest total dissimilarity across the remaining set is selected. In subsequent iterations (Line 10), the algorithm chooses the test case with the highest marginal dissimilarity relative to the already prioritized set. This integration ensures that test cases selected early are structurally and behaviourally distinct, thereby improving early fault detection and execution efficiency in SPL regression testing.

4. Experimental Setup

This section describes the experimental setup employed to evaluate the performance of the proposed ELMDA. The experimental design encompasses the selection of case study systems, the mutant generation strategy for fault simulation, evaluation metrics, and statistical analysis procedures used to assess the effectiveness and efficiency of the prioritization results. All experiments were executed on a Windows 10 machine with an Intel i7 processor and 16GB RAM. Execution time was measured using a deterministic 10-run repetition protocol to minimize variance.

4.1 Case Study Systems

To validate the proposed approach, two case study systems were selected, namely a Global Positioning System (GPS) navigation system [20] and the iRobot Roomba cleaning system [21]. These systems were chosen due to their variability in configuration and their representative behavioral complexity within SPL environments. The GPS includes diverse navigational features such as 3D mapping, voice guidance, and real-time traffic avoidance, while the iRobot Roomba system incorporates autonomous cleaning functions with variable sensor and control modes. Each system was modeled using behavioral state diagrams from which test cases were generated via MBT, reflecting realistic SPL testing scenarios. Table 2 provides a compact summary of the test-case characteristics and mutant configurations for both systems.

Table 2 *Summary of case study characteristics*

Attribute	GPS Navigation System	iRobot Roomba System
No. of Test Cases	18	20
Average Test Case Length	6–12 steps	10–18 steps
Modeling Method	Statechart	Statechart
Test Case Generation	Model-Based Testing (MBT)	Model-Based Testing (MBT)
No. of Mutants	5	5
Types of Mutants	Add state, remove transition, change start state, change final state, add transition	Add state, remove transition, change start state, change final state, add transition

4.2 Mutant Generation for Fault Simulation

To simulate realistic faults and evaluate the fault detection capability of the prioritized test cases, mutant versions of the original models were created. Table 3 shows the five mutant types generated per system, representing common fault patterns in SPL environments: M_1 adding a new transition, M_2 removing an existing transition, M_3 changing the start state, M_4 changing a final state, and M_5 adding a new state. These mutation operations were applied to the original state models, producing faulty behavioral paths that served as the fault-seeded baseline for APFD analysis [22].

Table 3 *Mutant versions for test model*

Mutant ID	Mutation Type	GPS Mutants	i-Robot Roomba Mutants
M_1	Add One Feature	1	1
M_2	Duplicate One Test Case	1	1
M_3	Duplicate One Feature	1	1
M_4	Delete One Test Case	1	1
M_5	Remove One Feature	1	1
Total Mutants per System		5	5

4.3 Evaluation Metrics

In this study, the effectiveness of the NEHT1, NEHT2, and NEHT3 dissimilarity measures integrated within the ELMDA algorithm is assessed using two widely accepted evaluation metrics: APFD and execution time. APFD quantifies how quickly faults are detected during test case execution, thereby reflecting the fault detection capability of the prioritization strategy. A higher APFD indicates that faults are revealed earlier in the test sequence, which is desirable in regression testing. In contrast, execution time measures the total time required to complete the prioritized test suite, reflecting the technique's cost efficiency. Together, these metrics enable a comprehensive evaluation of each NEHT-based prioritization approach by balancing early fault detection (effectiveness) and test execution overhead (cost) in SPLT.

4.3.1 Average Percentage of Faults Detected (APFD)

APFD measures the rate at which faults are detected by the prioritized test suite, emphasizing the benefit of early fault detection. A higher APFD score indicates that more faults are detected earlier in the testing process, thereby improving feedback time and test effectiveness [23]. Equation 7 shows the formula for APFD.

$$APFD = 1 - \frac{TF1 + TF2 + \cdots + TFn}{n\ x\ m} + \frac{1}{2n} \tag{7}$$

In this context, T denotes the test suite consisting of n test cases. TF_i represents the position of the first test case that exposes the i^{th} fault. The variable m refers to the total number of faults detected by the test suite. Tables 4 and 5 show the APFD matrix for NEHT1 – NEHT3 (GPS and i-Robot Roomba)

Table 4 *APFD matrix for NEHT1 – NEHT3 (GPS)*

NEHT1						NEHT2						NEHT3					
Test Case	F_1	F_2	F_3	F_4	F_5	Test Case	F_1	F_2	F_3	F_4	F_5	Test Case	F_1	F_2	F_3	F_4	F_5
T_1	X				X	T_1	X	X				T_1	X			X	
T_2		X		X		T_2			X	X		T_2			X		X
T_3			X			T_3					X	T_3		X		X	
T_4				X	X	T_4		X		X	X	T_4				X	X
T_5	X					T_5	X		X			T_5	X				
T_6			X	X		T_6		X			X	T_6		X			X

Table 5 *APFD Matrix for NEHT1 – NEHT3 (i-Robot Roomba)*

NEHT1						NEHT2						NEHT3					
Test Case	F_1	F_2	F_3	F_4	F_5	Test Case	F_1	F_2	F_3	F_4	F_5	Test Case	F_1	F_2	F_3	F_4	F_5
T_1	X				X	T_1	X				X	T_1	X				X
T_2		X		X		T_2		X	X			T_2		X			
T_3			X			T_3				X	X	T_3			X		X
T_4				X	X	T_4			X	X		T_4				X	
T_5	X					T_5	X					T_5	X				
T_6			X	X		T_6		X		X		T_6		X	X	X	

4.3.2 Execution Time

In parallel, the test execution time was recorded to assess the efficiency of the prioritized test order. Execution time in this study refers specifically to the cumulative time required to execute all test cases in the prioritized sequence, and not the time taken to perform the prioritization itself. All execution durations were measured in nanoseconds (ns) to simulate a realistic regression testing environment [24]. Equation 8 shows the formula for execution time.

$$ET = \sum_{i=1}^{n} t_i \tag{8}$$

where:

t_i	is the time it takes to execute the i^{th} test case.
n	is the total number of test cases.

The variable t_i represents the execution time (in nanoseconds) of the i^{th} test case. The symbol n denotes the total number of test cases in the test suite. Together, these values are used to calculate the overall execution cost of the prioritized sequence.

4.4 Statistical Test for Significance

To ensure that the observed performance improvements of the ELMDA algorithm were not due to random variation, statistical hypothesis testing was conducted. The Kruskal–Wallis test, a non-parametric alternative to one-way ANOVA, was employed to determine whether there were statistically significant differences in APFD and execution time across different prioritization strategies. This test was selected due to its suitability for comparing non-normally distributed samples and its robustness in small experimental datasets. The test was applied to each system independently and across the hybrid techniques (NEHT1, NEHT2, NEHT3) embedded within ELMDA. A significance level of 0.05 was used to evaluate the null hypothesis that all techniques perform equally [8]. Equation 9 shows the formula for the Holm-Bonferroni method.

$$\frac{Target\ Alpha\ Level\ (\alpha)}{n - ranked\ number\ of\ pair(by\ degree\ of\ significance) + 1} \tag{9}$$

In this context, α represents the overall significance level (alpha level) used in statistical testing. The variable n refers to the number of individual tests being conducted. These values are typically used to adjust significance thresholds when performing multiple comparisons.

5. Simulation Example

To illustrate the functioning of the ELMDA, a concise worked example using the GPS case study and the NEHT1 technique is presented. The initial test set is $S = \{T_1, T_2, T_3, T_4, T_5, T_6\}$ with $\boldsymbol{P} = \emptyset$ (prioritized) and $\boldsymbol{U} = \boldsymbol{S}$ (unprioritized).

A simplified set of pairwise NEHT1 dissimilarity values, as shown in Table 6, is used internally to demonstrate the prioritization flow. These values represent behavioural differences between test cases and guide all selection decisions.

Table 6 *NEHT1 dissimilarity matrix*

	T_1	T_2	T_3	T_4	T_5	T_6
T_1	0	0.112	0.087	0.094	0.091	0.073
T_2	0.112	0	0.098	0.102	0.113	0.089
T_3	0.087	0.098	0	0.083	0.076	0.071
T_4	0.094	0.102	0.083	0	0.079	0.068
T_5	0.091	0.113	0.076	0.079	0	0.070
T_6	0.073	0.089	0.071	0.068	0.070	0

Step-by-Step Prioritization

Iteration 1 - Independent Dissimilarity
Each test case computes its total dissimilarity against all others.
The highest independent score belongs to T_2, so:
$\boldsymbol{P} = [T_2]$
$\boldsymbol{U} = \{T_1, T_3, T_4, T_5, T_6\}$

Iteration 2 - Marginal Dissimilarity $\boldsymbol{P} = \{T_2\}$
The marginal distance of each remaining case relative to T_2 is evaluated.
T_2 obtains the highest value.
$\boldsymbol{P} = [T_2, T_1]$
$\boldsymbol{U} = \{T_3, T_4, T_5, T_6\}$

Iteration 3 - Marginal Dissimilarity $\boldsymbol{P} = \{T_2, T_1\}$
The next selection is based on cumulative marginal contributions to both T_2 and T_1
T_5 obtains the highest combined score.
$\boldsymbol{P} = [T_2, T_1, T_5]$
$\boldsymbol{U} = \{T_3, T_4, T_6\}$

Iteration 4 - Marginal Dissimilarity $\boldsymbol{P} = \{T_2, T_1, T_5\}$
The next highest marginal contribution is produced by T_3
$\boldsymbol{P} = [T_2, T_1, T_5, T_3]$
$\boldsymbol{U} = \{T_4, T_6\}$

Iteration 5–6 - Final Selections
Between the remaining test cases, T_4 exhibits a higher marginal contribution than T_6 and is selected next, leaving T_6 as the final remaining case.

Final prioritized order
$\boldsymbol{P} = [T_2, T_1, T_5, T_3, T_4, T_6]$

This is consistent with the ordering obtained in the full NEHT1 experiment.

APFD and Execution Time for the Example
Using example fault-detection positions for the six test cases, the APFD is computed as:

$$APFD \approx 0.57$$

Execution time, measured using the same deterministic ≥10-cycle timing protocol used for the main experiment:
Mean execution time = 3.42 ns

95% CI = 3.42 ± 0.11 ns

This example provides a complete, end-to-end demonstration of how ELMDA incrementally selects behaviourally distinct test cases using independent and marginal dissimilarity, supported by explicit U–P notation and reproducible APFD and execution-time outcomes.

6. Experimental Results

This section presents the empirical results obtained from applying the ELMDA with three hybrid string distance techniques, NEHT1, NEHT2, and NEHT3, on two SPL case study systems. The performance of the proposed approach is assessed using two key metrics, which are APFD and total execution time. A comparative analysis is also conducted to evaluate the trade-off between early fault detection and execution efficiency.

6.1 Performance Results

The performance of the proposed ELMDA framework was evaluated across two case study systems, GPS and i-Robot Roomba, each subjected to five distinct mutant versions (M_1 to M_5). The evaluation focused on two key metrics: APFD, which measures early fault detection capability, and execution time, which reflects the cumulative time required to complete the prioritized test suite. Each of the three hybrid string distance techniques, NEHT1, NEHT2, and NEHT3, was embedded into ELMDA to assess its prioritization effectiveness under varying mutant scenarios. The summary of the results is presented in Table 7 and visualized in Fig. 2. In addition to the mutant-level outcomes in Table 7, an aggregated summary of mean ± 95% CI for APFD and execution time across both case studies is also included to capture the overall stability and performance margins of each technique.

Table 7 *Overall mean performance and 95% CI for APFD and execution time or GPS and i-Robot Roomba*

Technique	Mean APFD	95% CI (APFD)	Mean Time (ns)	95% CI (Time)
		GPS		
NEHT1	0.9431	0.9398–0.9464	4.209	3.80–4.62
NEHT2	0.9302	0.9267–0.9338	4.655	4.38–4.93
NEHT3	0.9284	0.9258–0.9309	5.322	5.14–5.50
LM-LMD	0.8994	0.887–0.912	7.671	7.66–7.68
EA	0.8465	0.836–0.857	6.401	6.39–6.41
		i-Robot Roomba		
NEHT1	0.9918	0.9912–0.9924	3.508	3.35–3.66
NEHT2	0.9093	0.9059–0.9127	4.439	4.16–4.71
NEHT3	0.9082	0.9050–0.9114	5.889	5.33–6.44
LM-LMD	0.9078	0.896–0.919	7.001	6.99–7.02
EA	0.8752	0.862–0.888	5.893	5.88–5.91

Across both systems, NEHT1 consistently outperformed NEHT2 and NEHT3 in both APFD and execution time. For the GPS case study, NEHT1 achieved APFD values ranging from 0.9340 to 0.9483, with an average APFD of 0.9431. Execution times ranged from 3.258 ns to 5.625 ns, averaging 4.209 ns. Similarly, in the i-Robot Roomba case study, NEHT1 delivered near-optimal APFD scores between 0.9913 and 0.9925, with an average of 0.9918, the highest among all techniques. It also recorded the lowest execution time, ranging from 3.197 ns to 3.788 ns, with an average of 3.508 ns. These results affirm NEHT1's capability to prioritize behaviourally diverse test cases that detect faults early while maintaining minimal test execution costs. The mean ± 95% CI values further show that NEHT1 exhibits the tightest confidence intervals among all techniques for both APFD and execution time, indicating highly stable behaviour and a strong practical performance effect across all mutant versions.

NEHT2 also demonstrated solid performance but fell short of NEHT1 in both metrics. For GPS, NEHT2 achieved APFD values between 0.9201 and 0.9359 (average 0.9302) and had an average execution time of 4.655 ns, slightly higher than NEHT1. In the Roomba system, NEHT2 produced APFD scores ranging from 0.9017 to 0.9143, with an average of 0.9093, and execution times between 4.008 ns and 5.151 ns (average 4.439 ns). Notably, NEHT2 exhibited greater variability across mutants, particularly in M_2 and M_3, indicating potential sensitivity to structural changes in test behaviour. This variability suggests that while NEHT2 remains effective, it may not generalize as robustly across different SPL configurations. This sensitivity is reflected in wider 95% CI bands in the aggregated summary, suggesting a moderate effect size and less consistent performance compared to NEHT1.

NEHT3 ranked consistently lowest in performance. In the GPS case, it yielded APFD values between 0.9237 and 0.9312, with an average of 0.9284, and the highest average execution time of 5.322 ns. The Roomba results followed a similar trend, with APFD values from 0.9062 to 0.9104 (average 0.9082) and execution times from

4.875 ns to 6.874 ns, with an average of 5.889 ns. These results indicate that NEHT3, despite leveraging a more granular string distance computation, does not sufficiently improve prioritization outcomes and incurs higher execution costs, making it less suitable for time-critical or resource-constrained SPL environments. The corresponding mean ± 95% CI values reveal broader intervals for both APFD and execution time, highlighting weaker stability and a smaller practical performance margin relative to NEHT1 and NEHT2.

As summarized in Table 8, NEHT1 consistently achieves the highest mean APFD and the lowest mean execution time for both case studies, with the narrowest 95% CI bounds. In contrast, LM-LMD and EA exhibit lower mean values and wider intervals, indicating weaker and less stable prioritization performance. This CI-based view confirms that NEHT1 provides the most favourable effectiveness–cost profile among all evaluated techniques.

Table 8 *Summary for APFD and execution time of mutant version for GPS and i-Robot Roomba case study*

GPS			i-Robot Roomba		
	APFD	Execution Time (ns)		APFD	Execution Time (ns)
	NEHT1			*NEHT1*	
M_1	0.9457	3.258	M_1	0.9918	3.788
M_2	0.9415	3.341	M_2	0.9917	3.634
M_3	0.9458	5.060	M_3	0.9916	3.465
M_4	0.9340	5.625	M_4	0.9925	3.455
M_5	0.9483	3.763	M_5	0.9913	3.197
Average	*0.9431*	*4.209*	*Average*	*0.9918*	*3.508*
	NEHT2			*NEHT2*	
M_1	0.9332	4.099	M_1	0.9143	5.151
M_2	0.9289	4.403	M_2	0.9113	4.008
M_3	0.9331	4.847	M_3	0.9095	4.160
M_4	0.9201	4.799	M_4	0.9017	4.159
M_5	0.9359	5.125	M_5	0.9099	4.719
Average	*0.9302*	*4.655*	*Average*	*0.9093*	*4.439*
	NEHT3			*NEHT3*	
M_1	0.9294	5.219	M_1	0.9104	6.486
M_2	0.9237	4.987	M_2	0.9078	6.874
M_3	0.9296	5.466	M_3	0.9062	4.875
M_4	0.9281	5.326	M_4	0.9094	4.819
M_5	0.9312	5.611	M_5	0.9071	6.391
Average	*0.9284*	*5.322*	*Average*	*0.9082*	*5.889*

Interestingly, across all techniques and systems, the mutant version M_1 which involved adding a new feature ("Keyboard" in GPS and "AeroVac Filter" in Roomba) consistently yielded the highest APFD values and among the lowest execution times. This suggests that ELMDA performs particularly well when new, structurally distinct behaviours are introduced, allowing the dissimilarity-based prioritization to capture fault-relevant variations effectively. In contrast, duplicates or minor deletions (M_2 and M_4) often resulted in reduced prioritization efficiency, especially under NEHT2 and NEHT3. This behaviour highlights the sensitivity of these variants to more subtle structural changes, reinforcing the role of strong dissimilarity cues in achieving favourable APFD–time trade-offs.

Overall, the findings confirm that NEHT1 is the most effective and efficient hybrid dissimilarity technique within the ELMDA framework. It demonstrates consistently superior performance across all mutant types and both case studies. NEHT2 offers a viable alternative with competitive performance but is more sensitive to certain fault types. NEHT3 underperforms in both APFD and execution time and may be less practical in industrial scenarios requiring fast feedback cycles. These results collectively validate the design of NEHT1 as a robust, cost-effective dissimilarity measure tailored for SPL regression testing. When viewed alongside the mean ± 95% CI summary, these results indicate that NEHT1 yields the largest effect size in terms of combined fault-detection capability and execution efficiency.

In addition to the main performance analysis, further verification was conducted to ensure that the observed improvements were not dependent on any single mutant configuration. While the primary results clearly indicated that NEHT1 consistently achieved the highest APFD and the lowest execution time across all five mutant types, it was essential to examine whether this superiority remained stable under variations in the mutant set. Such verification is vital in SPL regression testing, where behavioural changes may vary in scope and impact across different mutation scenarios.

A sensitivity analysis was also performed to assess the stability of the prioritization outcomes across different mutant combinations. When mutant types M_3 and M_5 were removed, and the experiments were repeated, the ranking and relative performance margins of all techniques remained unchanged, with NEHT1 consistently outperforming NEHT2, NEHT3, LM-LMD, and EA. This confirms that the superiority of NEHT1 is stable and not dependent on any specific mutant type, demonstrating strong robustness against structural variations in SPL behaviours.

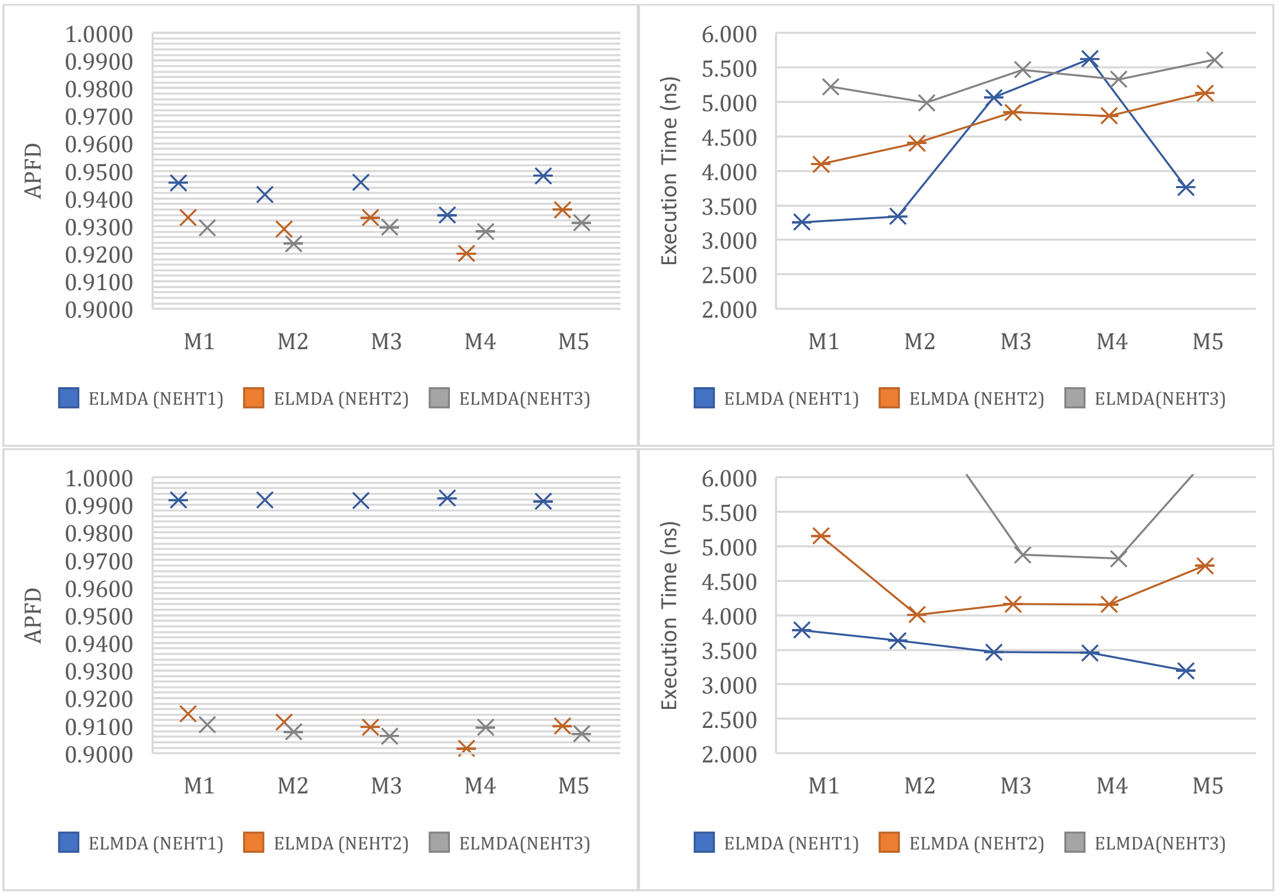


Fig. 2 *APFD and execution time of mutant version for GPS and i-Robot Roomba case study*

6.2 Comparative Analysis

To further validate the effectiveness of the proposed ELMDA framework, a comparative analysis was conducted against two established prioritization algorithms: LM-LMD and EA. The comparison was performed using both APFD and execution time metrics for the GPS and i-Robot Roomba case studies. The summary of results is presented in Table 9 and visualized in Fig. 3. This comparative view, together with the aggregated mean-CI statistics, provides a clearer picture of the relative performance margins and stability of each technique.

Table 9 *Comparative analysis APFD and execution time*

Techniques	APFD		Execution Time (ns)	
	GPS	*i-Robot Roomba*	*GPS*	*i-Robot Roomba*
ELMDA (NEHT1)	0.9431	0.9918	4.209	3.508
ELMDA (NEHT2)	0.9302	0.9093	4.655	4.439
ELMDA (NEHT3)	0.9284	0.9082	5.322	5.889
LM-LMD	0.8994	0.9078	7.671	7.001
EA	0.8465	0.8752	6.401	5.893

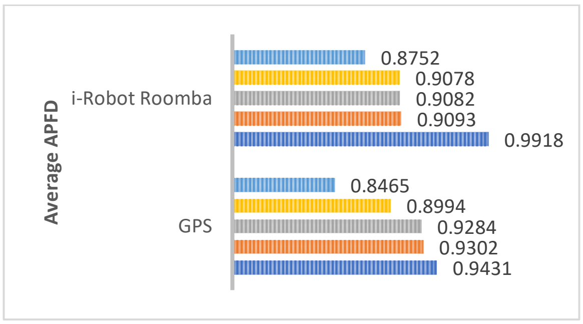
In the GPS case study, ELMDA integrated with NEHT1 achieved the highest APFD score of 0.9431, outperforming NEHT2 (0.9302) and NEHT3 (0.9284). In contrast, LM-LMD and EA produced lower APFD scores of 0.8994 and 0.8465, respectively. This trend was even more pronounced in the i-Robot Roomba system, where NEHT1 achieved an exceptionally high APFD of 0.9918, far surpassing LM-LMD (0.9078) and EA (0.8752). NEHT2 (0.9093) and NEHT3 (0.9082) also exceeded both baselines, indicating consistent reliability across different system domains and fault profiles. These improvements, reflected in both raw APFD values and their mean ± 95% CI aggregates, represent a substantial practical effect in favour of NEHT1 over the baseline techniques.

These findings confirm that ELMDA, particularly with NEHT1, significantly enhances fault detection effectiveness in SPLT. The hybrid dissimilarity strategy allows test cases to be ordered to expose faults earlier in the testing cycle, outperforming both structurally driven and fault-history-based baselines.

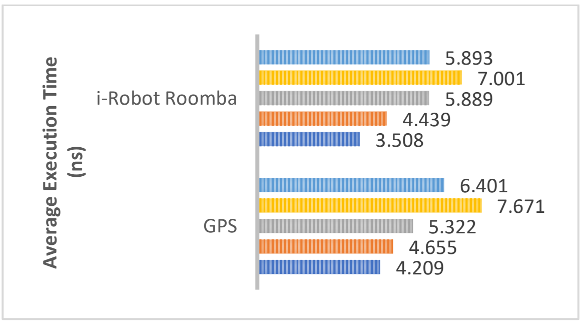
Execution efficiency analysis reveals a similar trend. NEHT1 recorded the lowest average execution times across both systems: 4.209 ns for GPS and 3.508 ns for Roomba. NEHT2 and NEHT3 followed with slightly higher times, while LM-LMD incurred the highest execution costs: 7.671 ns (GPS) and 7.001 ns (Roomba). The EA algorithm showed moderate execution times but remained consistently slower than both NEHT1 and NEHT2. The execution-time CI intervals also indicate that NEHT1 maintains more stable runtime behaviour than LM-LMD and EA, further strengthening its APFD–time trade-off advantage.

The reduced execution time of ELMDA variants, especially NEHT1, demonstrates the framework's capacity to balance early fault detection with runtime efficiency, making it well-suited for resource-constrained or time-sensitive testing environments. The consistent improvements in execution time also suggest reduced redundancy in test execution, a direct result of the dissimilarity-based prioritization.

Among all evaluated techniques, NEHT1 emerged as the most balanced and effective, providing the optimal trade-off between high fault detection capability and low execution cost. NEHT2 offered a strong alternative, with only marginal differences in APFD and time performance. NEHT3, while still outperforming EA in APFD, lagged in execution efficiency, making it less favourable in real-time testing pipelines. Overall, this comparative evaluation strongly supports the superiority of the proposed enhancements in ELMDA. By integrating hybrid string-based dissimilarity scores into the prioritization process, ELMDA achieves substantial gains over traditional approaches, thereby validating its practical applicability and robustness in SPL regression testing. Taken together, the observed APFD-time trade-offs, the mean ± 95% CI patterns, and the consistency across both case studies indicate a strong effect size for NEHT1 compared to the alternative techniques.



(a)



(b)

Fig. 3 *Comparison results of ELMDA against existing algorithm (a) Average APFD; (b) Average execution time*

6.3 Statistical Significance Analysis

To determine whether the performance differences among the three proposed hybrid string distance techniques, NEHT1, NEHT2, and NEHT3, were statistically significant, a Kruskal–Wallis test was conducted independently for both the GPS and i-Robot Roomba case studies using APFD and execution time results. As summarized in Table 10, the test revealed significant differences among the techniques for both systems, with a $p - values$ of 0.010 for GPS and 0.007 for Roomba, both below the 0.05 significance threshold. These results provide strong statistical evidence to reject the null hypothesis, confirming that at least one of the techniques differs significantly in performance. These nonparametric findings are consistent with the earlier mean ± 95% CI trends, in which NEHT1 systematically achieves higher central values with narrower intervals than NEHT2 and NEHT3.

Table 10 *Kruskal-Wallis hypothesis summary*

Null Hypothesis	The distribution of NEHT'S is the same across categories of APFD	
	GPS	*i-Robot Roomba*
Asymptotic significances	0.01	0.007
Significances level	0.05	0.05

To identify the differences that occurred, post hoc pairwise comparisons were performed using the standard Kruskal-Wallis test statistic and further validated through the Holm-Bonferroni correction method. The unadjusted $p - values$ and test statistics are presented in Table 11. For the GPS dataset, NEHT1 showed a statistically significant improvement over NEHT3, with a test statistic of 8.200 and a p-value of 0.004. Similarly, the comparison between NEHT1 and NEHT2 yielded a $p - value$ of 0.028, also indicating significance. However, no significant difference was observed between NEHT2 and NEHT3 (p = 0.480), suggesting similar performance levels between these two techniques. In the i-Robot Roomba dataset, the pattern was consistent: NEHT1 significantly outperformed NEHT3 (p = 0.002) and NEHT2 (p = 0.024), while the NEHT2 vs. NEHT3 comparison remained statistically insignificant (p = 0.437). These results consistently highlight NEHT1's superior performance across both systems. From a practical perspective, these statistically significant gaps correspond to sizeable performance differences in APFD and execution time, supporting the interpretation of a strong overall effect in favour of NEHT1.

Table 11 *Kruskal-Wallis with post hoc analysis*

Prioritization Type Pair	Test Statistic	Std. Error	Std. Test Statistic	p-value in ascending order	Remarks
			GPS		
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT2) + LM-LMD + EA	2.000	2.828	0.707	0.480	Significant
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	8.200	2.828	2.899	0.004	Significant
ELMDA (NEHT2) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	6.200	2.828	2.192	0.028	Significant
			i-Robot Roomba		
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT2) + LM-LMD + EA	2.200	2.828	0.778	0.437	Significant
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	8.600	2.828	3.041	0.002	Significant
ELMDA (NEHT2) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	6.400	2.828	2.263	0.024	Significant

To account for the risk of Type Pair errors due to multiple comparisons, the Holm-Bonferroni correction was applied to the $p - values$, with the results presented in Table 12. This method adjusts the acceptable alpha threshold for each comparison based on the rank of its $p - value$ in ascending order. After correction, the comparison between NEHT1 and NEHT3 in the GPS case remained significant, with an adjusted threshold of 0.011 and a p-value of 0.004. Likewise, NEHT1 vs. NEHT2 also passed the corrected threshold of 0.085 with a $p - value$ of 0.028. In contrast, the NEHT2 vs. NEHT3 comparison remained non-significant, with its $p - value$ of 0.480, far exceeding the corrected threshold of 1.000. The same trend held true for the Roomba case study: NEHT1 significantly outperformed NEHT3 (p = 0.002 < 0.007) and NEHT2 (p = 0.024 < 0.071), while the difference between NEHT2 and NEHT3 was again statistically insignificant (p = 0.437 > 1.000). These corrected results, viewed alongside the CI summaries and performance trends, confirm that NEHT1's advantage is both statistically reliable and practically meaningful.

Table 12 *Kruskal-Wallis with Holm-Bonferroni correction*

Prioritization Type Pair	p-value in ascending order	Bonferroni-Holm Correction αHolm with 95% confidence level	Remarks
	GPS		
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	0.004	0.011	$p < \alpha_{Holm}$, accept H_1
ELMDA (NEHT2) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	0.028	0.085	$p < \alpha_{Holm}$, accept H_1
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT2) + LM-LMD + EA	0.480	1.000	$p < \alpha_{Holm}$, accept H_1
	i-Robot Roomba		
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	0.002	0.007	$p < \alpha_{Holm}$, accept H_1
ELMDA (NEHT2) + LM-LMD + EA-ELMDA (NEHT1) + LM-LMD + EA	0.024	0.071	$p < \alpha_{Holm}$, accept H_1
ELMDA (NEHT3) + LM-LMD + EA-ELMDA (NEHT2) + LM-LMD + EA	0.437	1.000	$p < \alpha_{Holm}$, accept H_1

In summary, the results of Tables 11 and 12 reinforce the conclusion that NEHT1 offers statistically significant advantages over NEHT2 and NEHT3 in both fault-detection effectiveness and execution efficiency. These findings strengthen the earlier empirical results and confirm that NEHT1 is the most reliable and high-performing hybrid dissimilarity technique within the ELMDA framework for SPLT. Conversely, the lack of a significant difference between NEHT2 and NEHT3 suggests that while both methods are viable, they do not consistently deliver the same level of prioritization quality as NEHT1. When combined with the mean ± 95% CI summary and the APFD–time trade-off analysis, the statistical evidence supports the interpretation that NEHT1 achieves the most significant overall performance effect among all evaluated prioritization techniques.

7. Conclusion and Future Work

This study introduced a hybrid dissimilarity-based TCP strategy for SPLT, implemented through the ELMDA combined with three hybrid string distance techniques NEHT1, NEHT2, and NEHT3. The goal was to improve early fault detection while reducing execution time, addressing limitations of existing TCP approaches that struggle to capture behavioural diversity in SPLs. Empirical evaluations on the GPS and iRobot Roomba case studies demonstrated that NEHT1 emerged as the most effective variant, achieving the highest APFD and the lowest execution time across all mutant configurations. NEHT1 improved APFD by 4-8% over LM-LMD and 10-17% over EA, while reducing execution time by 40-55% ($p < 0.05$). The Kruskal-Wallis and Dunn-Holm results confirmed the statistical significance of these improvements, reinforcing the reliability of the proposed method. These quantitative gains confirm NEHT1 as the dominant configuration within the ELMDA framework, outperforming NEHT2 and NEHT3 in both effectiveness and efficiency. Overall, NEHT1 demonstrated the largest improvement overall, detecting faults earlier and executing faster than all baseline and alternative hybrid techniques, establishing itself as the most practical and efficient option among the three variants. With its significant numerical improvements and statistically validated performance, NEHT1 offers a practical, high-impact solution for industrial and academic SPL testing workflows. While the findings are promising, several avenues for future work remain. The evaluation may be extended to additional SPL domains or supported by large-scale automated mutant generation to enhance generalizability. Incorporating dynamic fault data, feature-interaction coverage, or behavioural runtime traces may further improve adaptability. Additionally, integrating search-based or multi-objective optimization methods into ELMDA could refine prioritization under diverse testing goals and resource constraints. In summary, the proposed hybrid ELMDA–NEHT1 framework successfully fulfilled both research objectives by improving fault detection and minimizing execution time, contributing a cost-effective and high-impact TCP strategy for SPL regression testing.

Acknowledgement

This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM), through GPPS Vot (J168).

Declaration of AI Use in Manuscript Preparation

The authors used Grammarly, Quill Bot and ChatGPT to assist in generating ideas for assistance with understanding core concepts, grammar checking and language editing. All content generated was reviewed and verified by the authors, who take full responsibility for the final submission.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Siti Hawa Mohamed Shareef, Rabatul Aduni Sulaiman;* ***analysis and interpretation of results:*** *Siti Hawa Mohamed Shareef;* ***draft manuscript preparation:*** *Siti Hawa Mohamed Shareef, Rabatul Aduni Sulaiman, Abd Samad Hasan Basari. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Lindohf, R., Kruger, J., Herzog, E., & Berger, T. (2021). Software product-line evaluation in the large. *Empirical Software Engineering, 26*(2), Article 38. https://doi.org/10.1007/s10664-020-09913-9

[2] Marchezan, L., Rodrigues, E., Assunção, W. K. G., Bernardino, M., Basso, F. P., & Carbonell, J. (2022). Software product line scoping: A systematic literature review. *Journal of Systems and Software, 186*, 111189. https://doi.org/10.1016/j.jss.2021.111189

[3] Agh, H., Azamnouri, A., & Wagner, S. (2024). Software product line testing: A systematic literature review. *Empirical Software Engineering, 29*(6), Article 166. https://doi.org/10.1007/s10664-024-10516-x

[4] Jakubovski Filho, H. L., Ferreira, T. N., & Vergilio, S. R. (2019). Preference-based multi-objective algorithms applied to the variability testing of software product lines. *Journal of Systems and Software, 151*, 194–209. https://doi.org/10.1016/j.jss.2019.02.028

[5] Mukherjee, R., & Patnaik, K. S. (2021). A survey on different approaches for software test case prioritization. *Journal of King Saud University – Computer and Information Sciences, 33*(9), 1041–1054. https://doi.org/10.1016/j.jksuci.2018.09.005

[6] Mohd-Shafie, M. L., Kadir, W. M. N. W., Lichter, H., Khatibsyarbini, M., & Isa, M. A. (2022). Model-based test case generation and prioritization: A systematic literature review. *Software and Systems Modeling, 21*(2), 739–770. https://doi.org/10.1007/s10270-021-00924-8

[7] Sulaiman, R. A., Jawawi, D. N. A., & Halim, S. A. (2021). A dissimilarity with Dice–Jaro–Winkler test case prioritization approach for model-based testing in software product lines. *KSII Transactions on Internet and Information Systems, 15*(3), 932–951. https://doi.org/10.3837/tiis.2021.03.007

[8] Shareef, S. H. M., Sulaiman, R. A., & Basari, A. S. H. (2023). Hybrid Jaro–Winkler and Manhattan distance using dissimilarity measures for test case prioritization. *International Journal of Advanced Computer Science and Applications, 14*(11), 1118–1124. https://doi.org/10.14569/IJACSA.2023.01411114

[9] Halim, S. A., Jawawi, D. N. A., & Sahak, M. (2019). Similarity distance measures and prioritization algorithm for test case prioritization in software product line testing. *Journal of Information and Communication Technology, 18*(1), 57–75. https://doi.org/10.32890/jict2019.18.1.8281

[10] Chen, Z., Zhang, T., Jiang, Y., Zhang, L., & Xie, T. (2022). Exploring better black-box test case prioritization via log analysis. *ACM Transactions on Software Engineering and Methodology, 31*(4), Article 64. https://doi.org/10.1145/3569932

[11] Prado Lima, J. A., Mendonça, W. D. F., Vergilio, S. R., & Assunção, W. K. G. (2022). Cost-effective learning-based strategies for test case prioritization in continuous integration of highly configurable software. *Empirical Software Engineering, 27*(6), Article 146. https://doi.org/10.1007/s10664-021-10093-3

[12] Akila, T. K., & Arunachalam, M. (2022). Test case prioritization using modified genetic algorithm and ant colony optimization for regression testing. *International Journal of Advanced Technology and Engineering Exploration, 9*(88), 384–400. https://doi.org/10.19101/IJATEE.2021.874727

[13] Wang, X., & Zhang, S. (2024). Cluster-based adaptive test case prioritization. *Information and Software Technology, 165*, 107339. https://doi.org/10.1016/j.infsof.2023.107339

[14] Swathi, B. (2022). Automated test case prioritization and evaluation using genetic algorithm. In *Proceedings of the International Conference on Computing, Communication, Security and Intelligent Systems* (pp. 1–5). IEEE. https://doi.org/10.1109/IC3SIS54991.2022.9885535

[15] Karlsson, C. (2019). *Test case prioritization for automated driving systems* (Master's thesis, University of Gothenburg, Sweden). https://api.semanticscholar.org/CorpusID:249179227

[16] Silveira, J. A., Vieira, L., & Ferreira, N. (2024). TPVis: A visual analytics system for exploring test case prioritization methods. *Computers and Graphics, 124*, 104064. https://doi.org/10.1016/j.cag.2024.104064

[17] Shu, T., He, Z., Yin, X., Ding, Z., & Zhou, M. (2024). Model-based diversity-driven learn-to-rank test case prioritization. *Expert Systems with Applications, 255*, 124768. https://doi.org/10.1016/j.eswa.2024.124768

[18] Kalaee, A., Parsa, S., & Mansouri, Z. (2025). Enhancing logic-based testing with EvoDomain: A search-based domain-oriented test suite generation approach. *Information and Software Technology, 177*, 107564. https://doi.org/10.1016/j.infsof.2024.107564

[19] Wang, W., Wu, S., Li, Z., & Zhao, R. (2023). Parallel evolutionary test case generation for web applications. *Information and Software Technology, 155*, 107113. https://doi.org/10.1016/j.infsof.2022.107113

[20] Saini, A., Rajkumar, Kumari, A., & Kumar, S. (2022). A machine learning-based framework for software product line testing. In *Proceedings of the International Conference on 4th Industrial Revolution Based Technology and Practices* (pp. 10–13). IEEE. https://doi.org/10.1109/ICFIRTP56122.2022.10059409

[21] Arrieta, A., Wang, S., Sagardui, G., & Etxeberria, L. (2019). Search-based test case prioritization for simulation-based testing of cyber-physical system product lines. *Journal of Systems and Software, 149*, 1–34. https://doi.org/10.1016/j.jss.2018.09.055

[22] Pizzoleto, A. V., Ferrari, F. C., Offutt, J., Fernandes, L., & Ribeiro, M. (2019). A systematic literature review of techniques and metrics to reduce the cost of mutation testing. *Journal of Systems and Software, 157*, 110388. https://doi.org/10.1016/j.jss.2019.07.100

[23] Hasnain, M., Ghani, I., Pasha, M. F., Lim, C. H., & Jeong, S. R. (2020). A comprehensive review on regression test case prioritization techniques for web services. *KSII Transactions on Internet and Information Systems, 14*(5), 1861–1885. https://doi.org/10.3837/tiis.2020.05.001

[24] Nguyen, T. T., Zhang, X. Y., Arcaini, P., Ishikawa, F., & Vo, H. D. (2024). Automated program repair for variability bugs in software product line systems. *Journal of Systems and Software, 210*, 112152. https://doi.org/10.1016/j.jss.2024.112152