# The Hybrid Jaro-Winkler and Manhattan Distance using Dissimilarity Measure for Test Case Prioritization Approach

Siti Hawa Mohamed Shareef, Rabatul Aduni Sulaiman*, Abd Samad Hasan Basari
Department of Software Engineering, Faculty of Computer Science and Information Technology,
University Tun Hussein Onn Malaysia, Batu Pahat, Malaysia

***Abstract*—Software product line (SPL) is a concept that has revolutionized the software development industry. It refers to a set of related software products that are developed from a common set of core assets but can be customized to meet specific customer requirements. Integrating SPL techniques into test case prioritization (TCP) can greatly enhance its effectiveness. By considering variability across different products within an SPL, it becomes possible to prioritize test cases based on their relevance to specific product configurations. However, the concept itself still has certain issues, such as in finding the highest rate of early failure detection. Various solutions have been proposed to mitigate this problem, among them is to improve the calculation of string distance using hybrid technique to achieve a high degree for similarity. Dissimilarity-based Technique (DBP) is the basis for our ranking method. The objective is to identify further weaknesses in the product lines as well as the differences between the experiment and real-world applications. Our focus is to enhance hybrid techniques that produce the highest rate of early failure detection. In this paper, early fault detection is selected as the performance goal. In order to choose the optimal methods for DBP for TCP, a comparison between several string distance measures was conducted. This study proposed hybrid techniques that combined Jaro-Winkler and Manhattan string distance namely New Enhanced Hybrid Technique 1 (NEHT1), New Enhanced Hybrid Technique 2 (NEHT2) and New Enhanced Hybrid Technique 3 (NEHT3). The case study was generated using the PLEDGE tool based on a Feature Model (FM). Six test cases were used in the experiment. Result shows the effectiveness of the combination where it achieved higher degree of similarity for T1 vs. T4, T2 vs. T3, T2 vs. T6, and T3 vs. T6, as well as perfect degree of similarity for NEHT1 (100.00%). The result proves that the combination of both techniques improve SPL testing effectiveness compared to existing techniques.***

***Keywords*—Test case prioritization; software product line; dissimilarity-based technique; string distance; new enhanced hybrid***

## I. INTRODUCTION

Software product line (SPL) is a collection of related software products that share a common set of core assets while also offering variations to address diverse customer needs [1]. The characteristics may be constant throughout all SPL-derived products, or they may be varied and present in only some of them [2]. Instead of building each product from scratch, SPL approach emphasizes systematic reuse, enabling efficient development and maintenance of multiple products. SPL streamlines development, reduces redundancy, and enhances consistency across products [3]. Many industries implement SPL due to its ability to handle different phases of development using the commonality and variability concepts [4].

Software product line testing (SPLT) involves testing the shared components and individual product variants within an SPL [5]. It ensures the quality, compatibility, and correctness of both the common core assets and the unique features of each product. This type of testing addresses the challenges posed by varying configurations, shared components, and differing features, while maintaining overall product line quality [6]. Similar to testing in non-configurable code, testing in SPL experiences the coincidental correctness phenomena, which makes it more challenging to detect errors in these systems [7]. However, the testing of a single software system is a highly difficult and expensive stage of the software development process, according to the author [8].

Test case prioritization (TCP) is the process of ordering test cases based on certain criteria to optimize testing efforts [9]. Even though a few trailing test cases are not exercised, these test suites uncover bugs at the earliest possible time [10]. It presents a significant difficulty for software testing [11]. In the context of SPL, prioritization becomes complex due to the diversity of features and configurations [12]. Researchers suggested TCP procedures, where test cases were restructured and carried out in accordance with a given objective, to boost the efficacy and efficiency of testing [13]. A hybrid approach that considers both similarity and dissimilarity should be adopted for effective TCP in SPL development. Similarity refers to the degree to which two or more test cases share common characteristics or requirements while dissimilarity refers to the differences between test cases [14]. Techniques like Jaro-Winkler distance and Manhattan distance can be used to compare test cases for similarity, dependencies, and impact. Prioritizing test cases ensures that critical defects are identified early and that testing resources are allocated efficiently.

SPL has gained prominence in modern software development by allowing the creation of multiple products with shared features and components. TCP plays a vital role in ensuring the quality and reliability of these products. Hybrid string distance, a combination of various string similarity metrics, presents a promising approach to enhancing TCP in

SPL [15]. Hybrid string distance for TCP in SPL faces challenges such as diverse feature sets, low scalability, requires careful consideration of appropriate metrics, and low adaptability to dynamic changes [16]. Managing multiple product variants, selecting appropriate metrics, and ensuring the hybrid approach remains effective and adaptable to evolving requirements are essential for successful implementation. To optimize outcomes in TCP using hybrid string distance in SPL, several objectives should be pursued including comprehensive metric selection, feature-driven prioritization, and scalable algorithm design, adaptability to changes, and empirical validation and evaluation. The hybrid approach should consider specific product variant features, create a mechanism for prioritizing test cases based on these features, and ensure scalability without compromising performance. Finally, the approach should be tested on real-world SPLs to demonstrate improvements in TCP accuracy, coverage, and overall software quality.

This paper addresses the limitations of current TCP methods that struggle to accurately gauge the semantic similarity between test cases, resulting in less than optimal prioritization outcomes. To tackle this issue, the study poses a research question: "Which new hybrid technique can offer the highest early failure detection rate in TCP?"

The research introduces an innovative TCP approach that combines Hybrid Jaro-Winkler and Manhattan distance, integrating a dissimilarity measure. The main contribution lies in enhancing the precision and efficacy of TCP. This technique is used to overcoming the difficulties linked to precisely measuring semantic similarity. This method aims to advance software testing by significantly enhancing prioritization outcomes by providing a more robust and dependable approach for early failure detection in TCP.

The following section outlines the relevant literature. Section III present the proposed approach in detailed, incorporating the experimental settings and a combination of string distance measures whereas Section IV discuss on the results and discussion are presented, leading to the conclusions, in Section V.

## II. Related Work

Incorporating TCP techniques like reordering test cases based on fault detection rate can significantly enhance the effectiveness of software testing by enabling early fault detection [17]. In the context of TCP for SPL, the term string distance refers to the measurement of the similarity or dissimilarity of various strings that stand in for test cases [8]. With this method, test cases are ranked according to how distinctive or diverse they are from one another in terms of the testing functionality, or the areas of the code covered. String distance allows for better resource allocation by identifying redundant or overlapping test cases [18]. However, according to Halim et al. [1], neglecting string distance would result in inefficient testing processes and delayed bug fixes. Therefore, incorporating string distance in TCP is essential for efficient software development [19, 20].

Similarity-based prioritization (SBP) focuses on identifying test cases that are similar to each other based on certain criteria, such as code coverage or functionality [1, 21]. The idea behind SBP is that if one test case covers a particular aspect of the software, then similar test cases are likely to cover the same aspect as well [22]. This approach is effective in reducing redundancy in testing efforts by selecting a representative subset of test cases [23]. However, dissimilarity-based prioritization (DBP) considers the diversity among test cases. DBP aims to select a diverse set of test cases that covers different aspects of the software under test [23, 24]. By considering dissimilarities between test cases, this approach ensures comprehensive coverage and reduces the risk of missing critical defects [25].

Sulaiman et al. [25] suggested a measurement based on maximal distance of dissimilarity measure for SPL, which assures thorough coverage and lowers the possibility of overlooking important faults. The study is based on the test case generated from a statechart in comparison to current work, which is based on the FM in the context of the SPL domain. By increasing string distance and prioritizing based on similarity, Halim et al. [1] suggested rearranging test cases to increase the rate of problem identification. The work compared various string distance measures and prioritization algorithms in order to determine the best methods for similarity-based on hybridization of Jaro-Winkler and Hamming distance equation.

Fault detection has been improved in existing studies via the use of new and enhanced hybrid techniques for string distance equations. Recent work by Pospisil et al. [26] aimed to enhance adaptive random TCP for model-based test suites using original technique for Jaccard, Manhattan distance and similarity functions. All of the examined systems achieved improved fault detection performance as a result of the proposed improvement. Another study by Kumar et al. [9] employed Item-based Collaborative Filtering (ICF) to prioritize and decrease the number of products before testing. Hamming string distance was used to calculate the degree of similarity between products. Results of the study show that this approach was able to reduce test suite size. Compared to the works by Pospisil et al. [26] and Kumar et al. [9], the current study concentrated more on using a hybrid string distance method to determine the degree of dissimilarity and then locate the distance with the greatest similarity reading.

## III. Proposed Approach

The ranking method we use is based on dissimilarity. Our objective is to find further weaknesses in the product lines being evaluated as well as the point of difference between test case and real world. The study concentrates on the following research question:

RQ1: Which new enhanced hybrid technique produces the best early failure detection rate?

We start by outlining the conditions of our experiment before going on to describe the findings.

### *A. Experimental Settings*

The experiment was carried out on Windows 11 with an AMD Ryzen 5 5625U processor running at 2.30 GHz and 8GB of RAM. The authors developed a New Enhanced Hybrid Techniques (NEHT) by improving string distance using three

hybrid techniques to evaluate the comparability of similarity and dissimilarity measures. For the purpose of generating configuration and prioritizing processes, this technique's similarity and dissimilarity measures will be assessed using current Feature Model (FM), Software Product Line Online Tool (SPLOT) and Product Line EDitor and tests GEneration (PLEDGE) tools. In SPL, FM allows for the systematic representation and management of features, their dependencies, and variations across different products [27]. SPLOT is a web-based tool that allows users to create incredibly dynamic Ajax-based setup and reasoning user interfaces [28], while PLEDGE is an open-source tool that selects and prioritizes product configurations, maximizing the feature interactions covered [29]. In order to test the SPL, the author selects an FM for machine learning based on the Global Positioning System (GPS) created by Saini et al. 2023 [8] as in Fig. 1. Due to the fact that not all possible feature combinations are viable, feature diagrams are used to limit the variety of a product line. Based on the FM in Fig. 1, the *.xml* files will be produced using SPLOT. The *.xml* file will be used to generate the six test cases displayed in Table I after being run using PLEDGE. An ordered list of configurations is often the outcome of a sampling method.

### B. Hybrid of String Distance

The purpose of the proposed approach is to find dissimilarity between two test cases. Two strings distances were chosen to develop the proposed approach which is Jaro-Winkler and Manhattan distances. Jaro-Winkler distance is a string distance algorithm that measures the similarity between two strings [30]. It has been widely used in various fields, including TCP. Meanwhile, Manhattan distance is a popular metric used in TCP and works by first creating a matrix of all possible pairs of test cases [15]. It is used to measure the distance between two points on a grid-like system, where the distance is calculated by adding the absolute differences of those coordinates. In software testing, this metric helps prioritize test cases based on their proximity to each other.

The selection of the Hybrid Jaro-Winkler and Manhattan Distance Using Dissimilarity Measure for TCP Approach is grounded in its distinctive ability to address the challenges prevalent in existing TCP approaches. The hybrid nature of the chosen method combines the strengths of Jaro-Winkler and Manhattan string distance, offering a comprehensive solution for accurately capturing semantic similarity between test cases. The integration of a dissimilarity measure further enriches the approach, enhancing the precision of TCP. The decision to adopt this method is motivated by its potential to significantly improve prioritization results and contribute to more effective early failure detection in TCP.

By improving two string distance techniques, this method will produce a new hybrid technique that is precise in obtaining faster early failure detection rate. Fig. 2 describes the combinations of two string distance to develop the three new enhanced hybrid techniques. New enhanced hybrid technique 1 (NEHT1) modifies existing Jaro equation, and Manhattan equation replaces value of m and t with value of test cases *(T1, T2)*. New enhanced hybrid technique 2 (NEHT2) combines Jaro-Winkler and Manhattan equations, replaces m value with n value and t with value of test cases *(T1, T2)*, adds value of test cases, divides with n value and multiply with 1-dj. New enhanced hybrid technique 3 (NEHT3) combines Jaro and Manhattan equations where the formula replaces value of m with n and t with value of test cases *(T1, T2)*.

TABLE I. CONFIGURATIONS OF GPS FEATURE MODEL

| Test Case | Configuration |
|---|---|
| T1 | {GPS, Routing, Traffic Avoiding, Interface, Auto-rerouting, Screen, Touch} |
| T2 | {GPS, Routing, Radio, Interface, 3D Map, AM, FM, Digital, Keyboard, Screen, LCD} |
| T3 | {GPS, Routing, Traffic Avoiding, Radio, Interface, Auto-rerouting, AM, FM, Digital, Screen, LCD} |
| T4 | {GPS, Routing, Interface, 3D Map, Keyboard, Screen, LCD} |
| T5 | {GPS, Routing, Traffic Avoiding, Interface, 3D Map, Auto-rerouting, Screen, Touch} |
| T6 | {GPS, Routing, Radio, Interface, Auto-rerouting, AM, FM, Digital, Keyboard, Screen, Touch} |

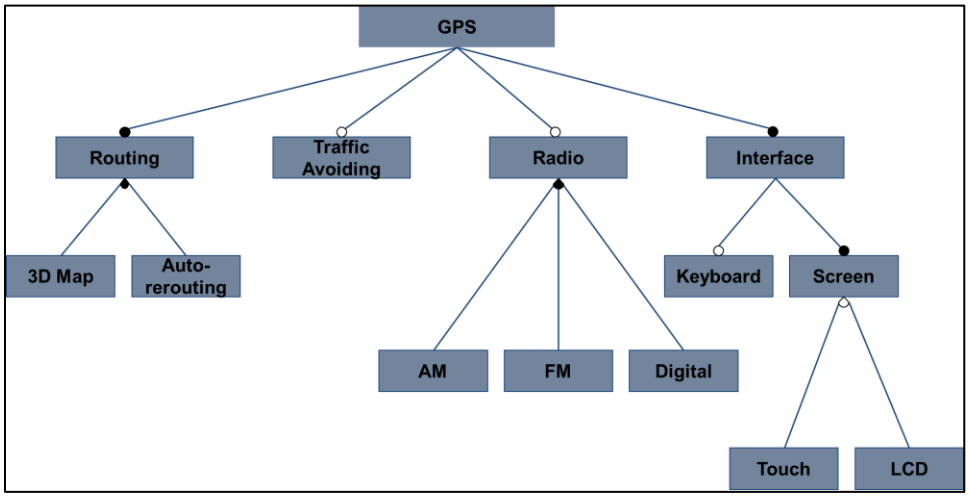


Fig. 1. Feature model of GPS [8].

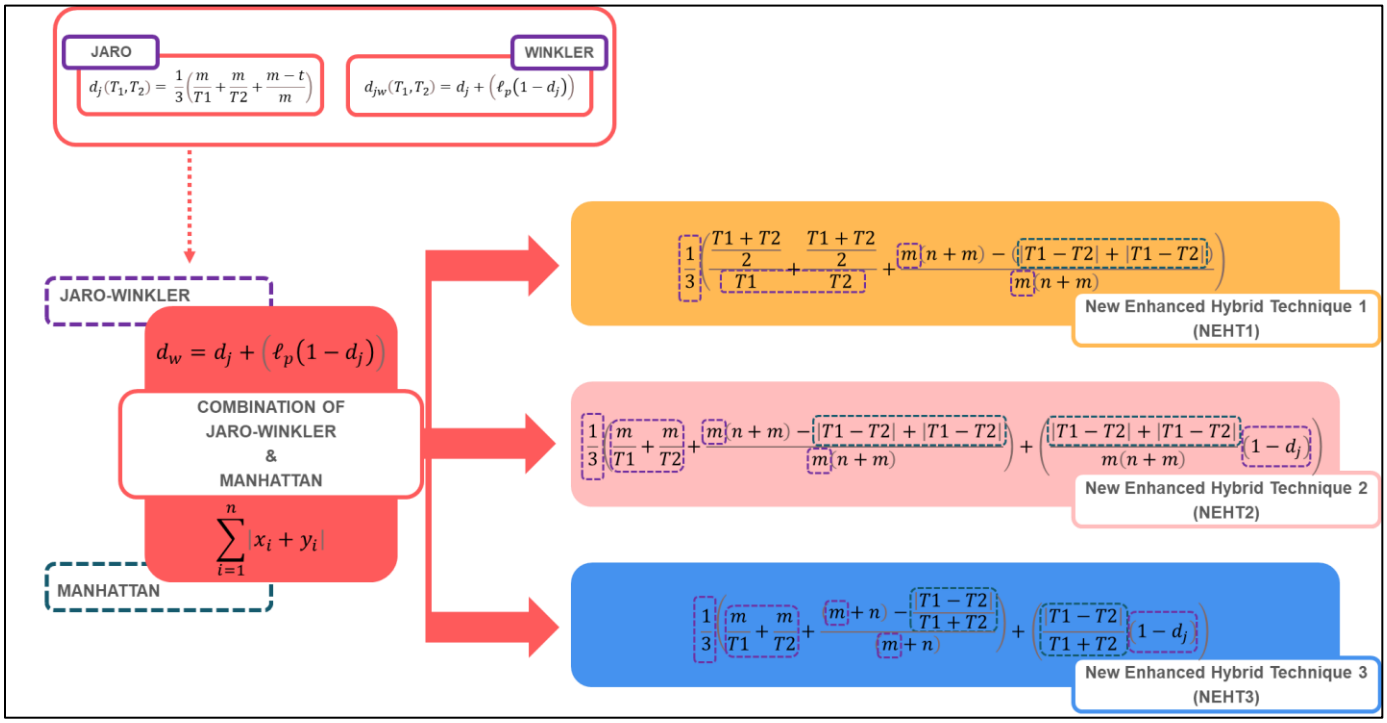


Fig. 2. New enhanced hybrid techniques.

## IV. Result and Discussion

Table II shows the similarity and dissimilarity percentages between different pairs of test cases *(T1, T2, T3, T4, T5, T6)*, with NEHT1, NEHT2, and NEHT3 representing different methods or conditions. The values range from 0% *(complete dissimilarity)* to 100% *(complete similarity)*. Since this is an initial result, results use a single FM to represent a dataset. For NEHT1, T1 vs T4, T2 vs T3, T2 vs T6, and T3 vs T6 recorded complete similarity (100.00%), proving the formula is very effective in similarity calculation. Values for NEHT2 and NEHT3 were similar in T1 vs. T4, T2 vs. T3, T2 vs. T6, and T3 vs. T6, which means both proposed techniques provide a consistent way to determine similarity level. The majority of the results show that NEHT1 is effective at determining the degree of similarity.

TABLE II. Calculation for Degrees of Similarity and Dissimilarity

| Test Case | NEHT1 | NEHT2 | NEHT3 | NEHT1 | NEHT2 | NEHT3 |
|---|---|---|---|---|---|---|
| | *Similarity (%)* | | | *Dissimilarity (%)* | | |
| T1 vs T2 | 86.79 | 82.25 | 70.95 | 13.21 | 17.75 | 29.05 |
| T1 vs T3 | 97.90 | 83.40 | 83.79 | 2.10 | 16.60 | 16.21 |
| T1 vs T4 | 100.00 | 71.42 | 71.42 | 0.00 | 28.58 | 28.58 |
| T1 vs T5 | 99.56 | 95.92 | 95.95 | 0.44 | 4.08 | 4.05 |
| T1 vs T6 | 97.90 | 83.40 | 83.79 | 2.10 | 16.60 | 16.21 |
| T2 vs T3 | 100.00 | 87.87 | 87.87 | 0.00 | 12.13 | 12.13 |
| T2 vs T4 | 99.65 | 89.26 | 89.99 | 0.35 | 10.74 | 10.01 |
| T2 vs T5 | 93.70 | 76.68 | 73.27 | 6.30 | 23.32 | 26.73 |
| T2 vs T6 | 100.00 | 87.87 | 87.87 | 0.00 | 12.13 | 12.13 |
| T3 vs T4 | 94.57 | 79.68 | 77.49 | 5.43 | 20.32 | 22.51 |
| T3 vs T5 | 96.94 | 79.87 | 79.59 | 3.06 | 20.13 | 20.41 |
| T3 vs T6 | 100.00 | 87.87 | 87.87 | 0.00 | 12.13 | 12.13 |
| T4 vs T5 | 98.81 | 78.95 | 79.21 | 1.19 | 21.05 | 20.79 |
| T4 vs T6 | 94.57 | 79.68 | 77.49 | 5.43 | 20.32 | 22.51 |
| T5 vs T6 | 96.94 | 79.87 | 79.59 | 3.06 | 20.13 | 20.41 |

Fig. 3 and Fig. 4 show similarity and dissimilarity rates of fault detection for the proposed methods *(NEHT1, NEHT2, NEHT3)*. The similarity percentages vary for different methods and test cases. Similar variations may be seen in the dissimilarity percentage, which illustrates how the different approaches of assessing differences differ. There is no uniform trend in how the methods rank similarity or dissimilarity across all test cases. Some test cases consistently show high similarity across all methods, while others show varying degrees of dissimilarity. The author claims that this enhancement will increase the SBP technique's effectiveness [1]. Sulaiman et al. [25] stated that similarity and dissimilarity strategies were introduced to tackle scalability problem in the current priority technique. This method provides a straightforward, scalable, and efficient method for prioritizing and reducing the number of test cases. TCP for SPL can be significantly improved by leveraging high similarity in calculation of string distance. High similarity values are advantageous in locating similar test cases across various SPL. As a result, fewer testing efforts are duplicated, and the existing test cases can be reused. Furthermore, low dissimilarity values can improve coverage, ensure effective bug correction, and improve the fault localization.

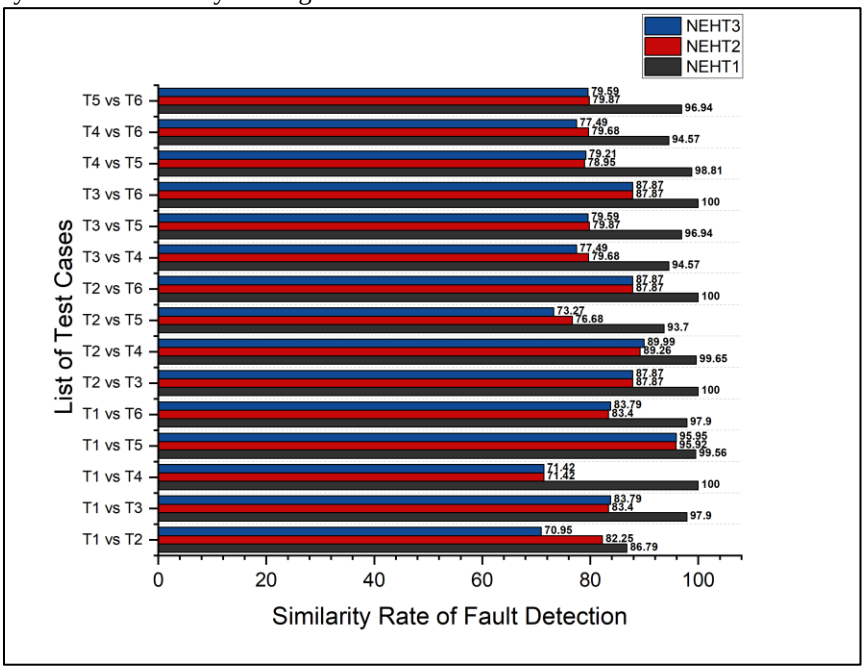


Fig. 3. Similarity rate of fault detection result.

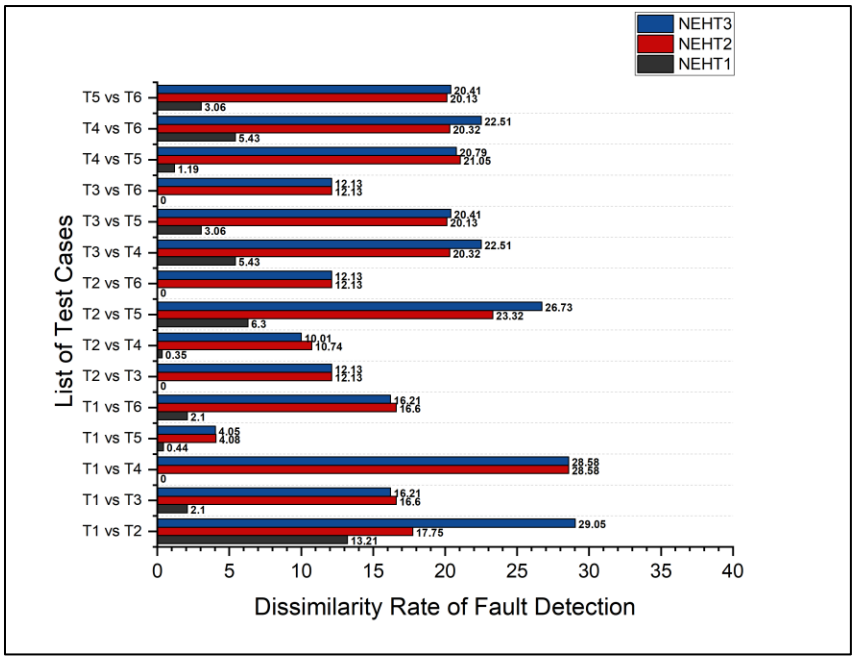


Fig. 4. Dissimilarity rate of fault detection result.

## V. CONCLUSION

One of the main advantages of SPLT is its ability to save time and resources. Testers can concentrate on the common characteristics shared by all products in the software family rather than testing each product individually. This makes it possible to quickly find and fix errors, at the same time shortens the development process and lowers expenses. For DBP, dissimilarity test case has been proven to be one of the techniques that can speed up failure detection process. This research employed six different test cases to be tested using three proposed hybrid techniques based on the combination of Jaro-Winkler and Manhattan string distances for early fault identification rate. The findings indicated that NEHT1 has a higher rate of fault identification compared to the other two proposed techniques. In order to increase the success rate of NEHT1's fault identification, we plan to make improvements to it in the future. In addition, we intend to use a variety of case study types for this research project. The limitations of current test case prioritization methods, particularly their struggles in accurately capturing semantic similarity, render them unsuitable for the challenges at hand. Traditional approaches fall short in providing a comprehensive solution for the nuanced characteristics of test cases. The proposed method is chosen to overcome these limitations by introducing a hybrid technique that specifically addresses the semantic aspects of test cases. This strategic choice is aimed at mitigating the deficiencies of existing methods and advancing the field of TCP towards a more precise and effective paradigm.

## ACKNOWLEDGMENT

This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme (FRGS/1/2022/ICT01/UTHM/03/2).

## REFERENCES

[1] S. A. Halim, D. N. A. Jawawi, and M. Sahak, "Similarity Distance Measure and Prioritization Algorithm for Test Case Prioritization in Software Product Line Testing," Journal of Information and Communication Technology, vol. 18, no. 1, pp. 57–75, 2019, doi: 10.32890/jict2019.18.1.8281.

[2] T. Ferreira, S. R. Vergilio, and M. Kessentini, "Variability testing of software product line: A preference-based dimensionality reduction approach," Inf Softw Technol, vol. 152, no. September 2021, p. 107031, 2022, doi: 10.1016/j.infsof.2022.107031.

[3] S. Langstrom, "An Investigative Study of Testing Strategy and Test Case Creation in a Hardware-Software Co-design Environment Using Software Product Line Theory," Open Access in DiVA, 2021.

[4] R. A. Sulaiman, D. N. A. Jawawi, and S. A. Halim, "Cost-effective test case generation with the hyper-heuristic for software product line testing," Advances in Engineering Software, vol. 175, Jan. 2023, doi: 10.1016/j.advengsoft.2022.103335.

[5] P. Martou, K. Mens, B. Duhoux, and A. Legay, "Test scenario generation for feature-based context-oriented software systems," Journal of Systems and Software, vol. 197, Mar. 2023, doi: 10.1016/j.jss.2022.111570.

[6] J. Lee, S. Kang, and P. Jung, "Test coverage criteria for software product line testing: Systematic literature review," Inf Softw Technol, vol. 122, no. December 2019, p. 106272, 2020, doi: 10.1016/j.infsof.2020.106272.

[7] T. T. Nguyen, K. T. Ngo, S. Nguyen, and H. D. Vo, "Detecting false-passing products and mitigating their impact on variability fault localization in software product lines," Inf Softw Technol, vol. 153, Jan. 2023, doi: 10.1016/j.infsof.2022.107080.

[8] A. Saini, Rajkumar, A. Kumari, and S. Kumar, "A Proposed Method of Machine Learning based Framework for Software Product Line Testing," 2022 International Conference on 4th Industrial Revolution Based Technology and Practices, ICFIRTP 2022, pp. 10–13, 2022, doi: 10.1109/ICFIRTP56122.2022.10059409.

[9] S. Kumar, Rajkumar, and M. Rani, "Collaborative Filtering-based Test Case Prioritization and Reduction for Software Product-Line Testing," in IEEE Region 10 Annual International Conference, Proceedings/TENCON, Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 498–503. doi: 10.1109/TENCON.2019.8929705.

[10] A. D. Shrivathsan et al., "Novel Fuzzy Clustering Methods for Test Case Prioritization in Software Projects," Symmetry (Basel), vol. 11, no. 11, Nov. 2019, doi: 10.3390/sym11111400.

[11] Z. Q. Zhou, C. Liu, T. Y. Chen, T. H. Tse, and W. Susilo, "Beating Random Test Case Prioritization," IEEE Trans Reliab, vol. 70, no. 2, pp. 654–675, 2021, doi: 10.1109/TR.2020.2979815.

[12] I. Hajri, A. Goknil, F. Pastore, and L. C. Briand, "Automating system test case classification and prioritization for use case-driven testing in product lines," Empir Softw Eng, vol. 25, no. 5, pp. 3711–3769, Sep. 2020, doi: 10.1007/s10664-020-09853-4.

[13] M. L. Mohd-Shafie, W. M. N. W. Kadir, H. Lichter, M. Khatibsyarbini, and M. A. Isa, "Model-based test case generation and prioritization: a systematic literature review," Softw Syst Model, vol. 21, no. 2, pp. 717–753, Apr. 2022, doi: 10.1007/s10270-021-00924-8.

[14] S. Akhmedova, V. Stanovov, and Y. Kamiya, "A Hybrid Clustering Approach Based on Fuzzy Logic and Evolutionary Computation for Anomaly Detection," Algorithms, vol. 15, no. 10, Oct. 2022, doi: 10.3390/a15100342.

[15] M. Khatibsyarbini, "A Study of Test Case Prioritization Technique Based on String Distance Metrics," Universiti Teknologi Malaysia, Johor Bahru, 2019.

[16] U. Markiegi, A. Arrieta, L. Etxeberria, and G. Sagardui, "Dynamic test prioritization of product lines: An application on configurable simulation models," Software Quality Journal, vol. 29, no. 4, pp. 943–988, Dec. 2021, doi: 10.1007/s11219-021-09571-0.

[17] T. K. Akila and M. Arunachalam, "Test case prioritization using modified genetic algorithm and ant colony optimization for regression testing," International Journal of Advanced Technology and Engineering Exploration, vol. 9, no. 88, pp. 384–400, Mar. 2022, doi: 10.19101/IJATEE.2021.874727.

[18] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed, and M. D. Mohamed Suffian, "Test Case Prioritization Using Firefly Algorithm for Software Testing," IEEE Access, vol. 7, pp. 132360–132373, 2019, doi: 10.1109/ACCESS.2019.2940620.

[19] S. Khoshmanesh and R. Lutz, "Does Link Prediction Help Detect Feature Interactions in Software Product Lines (SPLs)?," in Proceedings - 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020, 2020, pp. 87–90. doi: 10.1109/AIRE51212.2020.00020.

[20] C. Birchler, S. Khatiri, P. Derakhshanfar, S. Panichella, and A. Panichella, "Single and Multi-objective Test Cases Prioritization for Self-driving Cars in Virtual Environments," Proc ACM Hum Comput Interact, vol. 5, no. CSCW1, Apr. 2021, doi: 10.1145/1122445.1122456.

[21] S. Ali et al., "Towards Pattern-Based Change Verification Framework for Cloud-Enabled Healthcare Component-Based," IEEE Access, vol. 8, pp. 148007–148020, 2020, doi: 10.1109/ACCESS.2020.3014671.

[22] H. Hemmati, "Advances in Techniques for Test Prioritization," in Advances in Computers, Academic Press Inc., 2019, pp. 185–221. doi: 10.1016/bs.adcom.2017.12.004.

[23] S. Lity, M. Nieke, T. Thum, and I. Schaefer, "Retest test selection for product-line regression testing of variants and versions of variants," Journal of Systems and Software, vol. 147, pp. 46–63, Jan. 2019, doi: 10.1016/j.jss.2018.09.090.

[24] E. Ufuktepe and T. Tuglular, "Application of the law of minimum and dissimilarity analysis to Regression Test Case Prioritization," IEEE Access, vol. 11, pp. 57137–57157, 2023, doi: 10.1109/ACCESS.2023.3283212.

[25] R. A. Sulaiman, D. N. A. Jawawi, and S. A. Halim, "A Dissimilarity with Dice-Jaro-Winkler Test Case Prioritization Approach for Model-

Based Testing in Software Product Line," KSII Transactions on Internet and Information Systems, vol. 15, no. 3, pp. 932–951, Mar. 2021, doi: 10.3837/tiis.2021.03.007.

[26] T. Pospisil, J. Sobotka, and J. Novak, "Enhanced Adaptive Random Test Case Prioritization for Model-Based Test Suites," Acta Polytechnica Hungarica, vol. 17, no. 7, pp. 125–144, 2020, doi: 10.12700/APH.17.7.2020.7.7.

[27] M. Al-Hajjaji, T. Thum, M. Lochau, J. Meinicke, and G. Saake, "Effective product-line testing using similarity-based product prioritization," Softw Syst Model, vol. 18, no. 1, pp. 499–521, 2019, doi: 10.1007/s10270-016-0569-2.

[28] M. Mendonca, M. Branco, and D. Cowan, "S.P.L.O.T. - Software product lines online tools," Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, no. May 2014, pp. 761–762, 2009, doi: 10.1145/1639950.1640002.

[29] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. Le Traon, "PLEDGE: A Product Line Editor and Test Generation Tool," ACM International Conference Proceeding Series, pp. 126–129, 2013, doi: 10.1145/2499777.2499778.

[30] J. M. Keil, "Efficient Bounded Jaro-Winkler Similarity Based Search," Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), vol. P-289, pp. 205–214, 2019, doi: 10.18420/btw2019-13.