WEIGHTED STRING DISTANCE APPROACH BASED ON MODIFIED CLUSTERING TECHNIQUE FOR OPTIMIZING TEST CASE PRIORITIZATION

MUHAMMAD KHATIBSYARBINI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Doctor of Philosophy

Faculty of Computing
Universiti Teknologi Malaysia

OCTOBER 2022

DEDICATION

This thesis is dedicated to my beloved parents, who taught me that the best kind of knowledge to have is that to believe that everything come from Him and was destiny for specific purposed. It is also dedicated to my lecturers, who taught me that even the largest task can be accomplished if it is done one step at a time.

ACKNOWLEDGEMENT

Praise upon to HIM, for the blessings that I am able to finish this research. First of all, I would like to express my gratitude to my parents, my wife and family members for giving me the non-stop morale support for me to complete my studies.

In preparing this thesis, I was in contact with many people, researchers, academicians, and practitioners. They have contributed towards my understanding and thoughts. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Professor Dr. Mohd Adham bin Isa, for encouragement, guidance, critics and friendship. I am also very thankful to my co-supervisor Prof. Dr. Dayang Norhayati bt. Abang Jawawi for their guidance, advices and motivation. Without their continued support and interest, this thesis would not have been the same as presented here.

I am also indebted to Universiti Teknologi Malaysia (UTM) for funding my PhD study. My fellow postgraduate student should also be recognised for their support. My sincere appreciation also extends to all my colleagues and others who have provided assistance at various occasions. Their views and tips are useful indeed. Unfortunately, it is not possible to list all of them in this limited space.

ABSTRACT

Numerous test case prioritization (TCP) approaches have been introduced to enhance the test viability in software testing activity with the goal to maximize early average percentage fault detection (APFD). String based approach had shown that applying a single string distance-based metric to differentiate the test cases can improve the APFD and coverage rate (CR) results. However, to precisely differentiate the test cases in regression testing, the string approach still requires an enhancement as it lacks priority criteria. Therefore, a study on how to effectively cluster and prioritize test cases through string-based approach is conducted. To counter the string distances problem, weighted string distances is introduced. A further enhancement was made by tuning the weighted string metric with K-Means clustering and prioritization using Firefly Algorithm (FA) technique for the TCP approach to become more flexible in manipulating available information. Then, the combination of the weighted string distances along with clustering and prioritization is executed under the designed process for a new weighted string distances-based approach for complete evaluation. The experimental results show that all the weighted string distances obtained better results compared to its single string metric with average APFD values 95.73% and CR values 61.80% in *cstcas* Siemen dataset. As for the proposed weighted string distances approach with clustering techniques for regression testing, the combination obtained better results and flexibility than the conventional string approach. In addition, the proposed approach also passed statistical assessment by obtaining p-value higher than 0.05 in Shapiro-Wilk's normality test and p-value lower than 0.05 in Tukey Kramer Post Hoc tests. In conclusion, the proposed weighted string distances approach improves the overall score of APFD and CE and provides flexibility in the TCP approach for regression testing environment.

ABSTRAK

Banyak pendekatan keutamaan kes ujian (TCP) telah diperkenalkan untuk meningkatkan daya maju ujian dalam aktiviti pengujian perisian dengan matlamat untuk memaksimumkan pengesanan kesalahan peratusan purata awal (APFD). Pendekatan berasaskan rentetan telah menunjukkan bahawa dengan menggunakan metrik berasaskan jarak rentetan tunggal untuk membezakan kes ujian boleh meningkatkan keputusan APFD dan kadar liputan (CR). Walau bagaimanapun, untuk membezakan kes ujian dengan tepat dalam ujian regresi, pendekatan rentetan masih memerlukan peningkatan kerana ia tidak mempunyai kriteria keutamaan. Oleh itu, satu kajian tentang cara mengelompokkan dan mengutamakan kes ujian secara berkesan melalui pendekatan berasaskan rentetan dijalankan. Untuk mengatasi masalah jarak rentetan, jarak rentetan berwajaran diperkenalkan. Penambahbaikan selanjutnya dibuat dengan menala metrik jarak rentetan berwajaran dengan pengelompokan K-Means dan keutamaan menggunakan teknik algorithma kelip-kelip (FA) untuk pendekatan TCP agar menjadi lebih fleksibel untuk memanipulasi maklumat yang tersedia. Kemudian, gabungan metrik jarak rentetan berwajaran bersama pengelompokan dan keutamaan dilaksanakan di bawah proses yang direka bentuk untuk pendekatan berasaskan jarak rentetan berwajaran baharu untuk penilaian lengkap. Keputusan eksperimen menunjukkan bahawa semua metrik jarak rentetan berwajaran memperoleh hasil yang lebih baik berbanding dengan metrik rentetan tunggalnya dengan purata nilai APFD 95.73% dan nilai CR 61.80% dalam set data cstcas Siemen. Bagi pendekatan jarak rentetan berwajaran yang dicadangkan dengan teknik pengelompokan untuk ujian regresi, gabungan itu berjaya memperoleh hasil dan fleksibiliti yang lebih baik berbanding dengan pendekatan rentetan konvensional. Di samping itu, pendekatan yang dicadangkan juga melepasi penilaian statistik dengan mendapatkan nilai p lebih tinggi daripada 0.05 dalam ujian normaliti Shapiro-Wilk dan nilai p lebih rendah daripada 0.05 dalam ujian Post Hoc Tukey Kramer. Sebagai kesimpulan, pendekatan jarak rentetan berwajaran yang dicadangkan berjaya meningkatkan skor keseluruhan APFD dan CE serta menyediakan fleksibiliti dalam pendekatan TCP untuk persekitaran ujian regresi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligent
APFD	-	Average Percentage Fault Detection
ANOVA	-	Analysis of Variance
CR	-	Coverage Rate
CS	-	Cosine Similarity
GA	-	Genetic Algorithm
FA	-	Firefly Algorithm
LBS	-	Local Beam Search
LOC	-	Line of Code
ML	-	Machine Learning
NN	-	Nearest Neighbour Algorithm
PSO	-	Particle Swarm Optimization
RWS	-	Robotic Wheelchair System
SLR	-	Systematic Literature Review
SI	-	Swarm Intelligence
TCP	-	Test Case Prioritization
TF-IDF	-	Term Frequency – Inverse Document Frequency
TSP	-	Travelling Salesman Problem

LIST OF SYMBOLS

D	-	Distances
df	-	Data iteration frequency
F	-	Faults revealed
m	-	Total faults revealed
n	-	Total number of
T	-	Terms in all documents / test cases
t	-	Terms in single document / test case
TC	-	Test cases
TF	-	Sequence number of faults revealed
W	-	Weight

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

The first and foremost chapter presents the background of the study, challenges in the current Test Case Prioritization (TCP) methodology, challenges in test case distinction, challenges in the applied algorithm in TCP, motivation of study, problem statement, research objectives, the scope of the research and significance of the study. These challenges have been discussed in-depth to identify the research gaps that matched the research questions. Research objectives are then outlined to answer the research questions. The research boundaries are defined by outlining the research scopes. Later, the significance of the research is discussed to emphasize the importance and contribution of the study to enhance the existing approaches in the domain area.

Software engineering is not just programming and software development. Software engineering is the implementation of engineering procedures in the development of any software systematically. Within a software development process, software testing consumes a longer time in execution and can be the most expensive phase (Myers et al., 2004). Software testing is typically carried out even with time constraints and fixed resources (Mohd-Shafie et al., 2021). Software engineering groups are regularly compelled to end their testing activities because of financial and time necessities, which will trigger difficulties such as software quality and client agreement problems.

As the software evolves, a software test suite tends to increase in size, frequently making it expensive to execute. Research shows software testing is an expensive process that may require more than 33% of the cumulative expenses of the software (Chittimalli & Harrold, 2009; Zhang et al., 2018). In the work of Yoo and Harman (Pan et al., 2022), various test approaches were examined to supplement the importance of the accumulated test suite in testing. Those studies were then classified into three domains; test case minimization, test case selection, and test case prioritization.

Test case prioritization (TCP) aims to order a set of test cases to achieve an early optimization based on preferred properties (Amit Kumar & Singh, 2014). It gives an approach the ability to execute highly significant test cases first according to some measure and produce the desired outcome, such as revealing faults earlier and providing feedback to the testers. It also helps to find the ideal permutation of a series of test cases and could be executed accordingly (Mohd-Shafie et al., 2020; Yoo & Harman, 2012a). There are many dimensions of test case prioritization techniques. As many as eight broad dimensions were described by (Singh, 2012). Each approach has specified potential values, advantages, and limitations.

The inputs and dataset type play an essential role in the determination of their advantages and limitation. As there were different approaches, the methodology for each approach may also vary (Arora & Bhatia, 2018). As a result of this variation, it benefits project managers in adjusting their project schedules to counter the constraint within the project development process. However, most TCP approaches are not flexible enough to adequately fit different types of changes in regression testing (Garousi et al., 2018). This has resulted in less TCP utilization in current software development trends. Therefore, there is a gap in having a TCP approach that can adapt to different types of software development environment changes.

Artificial intelligence (AI) has emerged in TCP as one of the techniques that can support different types of changes but requires well-coded algorithms to cover multiple types of changes by itself. Regardless of the difficulty, the technique utilization in TCP is becoming the most popular TCP approach available nowadays. Numerous works have emerged recently (Bajaj & Sangwan, 2019; Hao, Zhang, Zang, et al., 2016; Prado & Vergilio, 2019; Shahbazi & Miller, 2016). AI techniques have been successfully used to reduce the effort of many software engineering activities (Hao, Zhang, Zang et al., 2016). In particular, machine learning (ML) techniques, a research field at the intersection of AI, computer science, and statistics, have been applied to automate various software engineering activities (Mece et al., 2020). In the TCP approach, the ML technique has been well welcomed recently (Mece et al., 2020). However, as software systems have become gradually complex, some conventional TCP approaches may not scale well to the complexity of these modern software systems. This snowballing complexity of recent software systems has solidified the neediness of ML techniques in TCP.

As the system was going to evolve, the process normally starts with requirement changes down to its model and, finally, the source code. These three phases of change hold helpful information that can be used as the resources and inputs for the TCP process. Each phase may have contributed a different level of information either as its reliability or precisely noted. All these changes can be clustered and classified to help the developers keep track of the software changes during the software developments. This can be an essential contributing factor to software project success, leading to less maintainable software. Thus, having a clustering technique in the TCP process is worth to be explored to achieve effective software testing by reducing test effort and increasing fault detection.

To apply ML techniques in TCP, each of the test cases available has to be distinguished properly. One of the methods is by using string metrics. String metrics play an important role in text-related research and applications in tasks such as information retrieval, text classification, document clustering, topic detection, topic tracking, questions generation, question answering, essay scoring, short answer scoring, machine translation, text summarization, and others (Gomaa & Fahmy, 2013). The string metric can be categorized based on its metric calculation, such as distances, similarity, and weight. However, to precisely calculate the distance between test cases, a specific and reliable string metric with specific priority was required where existing works (Bo Jiang & Chan, 2015; Ledru et al., 2012) have utilized one simple string similarity metric only, which produce numerous redundancy in equivalent value. Therefore, there is a need for enhancement of string similarity metric in order to overcome the issues.

To complete the TCP approach process, prioritization activity may occur after the clustering ML technique, which uses string metrics as discussed earlier. Recent works (Bo Jiang & Chan, 2015; Prado & Vergilio, 2019) indicate that the heuristic prioritization algorithm can significantly affect the TCP process. Their findings suggest that heuristic algorithms may increase the average percentage of fault detection (APFD) results. However, the result may differ if a clustering technique is applied before prioritization activity takes place. Therefore, there is a need for improvement to be made to prioritize the clustered test cases using string metric with its priority, improving the efficiency of the whole TCP process.

The main challenge to the problems alluded to can be divided into three primary issues: TCP Approaches that do not adequately address different types of changes, single string similarity metric formulation, and prioritization with ML technique. The string distances determine how far the distances between each test case are based on their attribute, such as test case inputs. ML techniques were then used to cluster the test cases, which must be prioritized to complete the TCP process. This prioritization algorithm is crucial to prioritize the clustered test cases with its calculated distance values among them. As for the process of TCP, it is meant to provide systematic guidance on how to execute the TCP process with an ML technique, specifically with consideration of string metrics. All the issues aim to address the issues of systematic fault detection in test case prioritization, problems which will be explained in detail in the following sections.

1.2 Challenge in Test Case Prioritization

Software engineering is highly concerned about systematically applying engineering processes to software development. Therefore, it is necessary to have a systematic process for the ML technique in the TCP approach. Numerous works exhibit similar process flow, with the only notable difference in adding or reducing one step to an existing process flow (Arora & Bhatia, 2018; Bajaj & Sangwan, 2019; Kazmi et al., 2017). This variation of process flow may give different results in a similar technique with the same dataset. However, a well-described approach using the combination of string similarity based on ML techniques has yet to be introduced

(Mukherjee & Patnaik, 2021). Therefore, the challenge in this process can be recognized as how to apply an ML technique in TCP with consideration of string metrics into the testing environment to improve the effectiveness of the process?

Generally, the TCP process starts with the preparation of data; even though no single paper clearly states it, it is compulsory for any experiment or research to identify which information or data will be used (Mukherjee & Patnaik, 2021). The data or information in TCP can be in the form of requirement statements, system models, and source code. The process is followed by determining and calculating prioritization criteria or dependency based on the data chosen. Then, the process goes on to prioritize the calculated criteria or dependency and measure the performance. However, formally defined steps and processes, especially in ML techniques with the challenges stated, are worth addressing (Bagherzadeh et al., 2021).

String metrics play an important role in text-related research and applications in tasks such as information retrieval, text classification, document clustering, topic detection, topic tracking, question generation, question answering, essay scoring, short answer scoring, machine translation, text summarization, and others. For example, the work by Jiang and Chan (Bo Jiang & Chan, 2015) tries to maximize test case diversity through test cases input information which is differ from Ledru's work (Ledru et al., 2012), where they treat each test case as a string of characters and prioritize test cases by using a simple string edit distance to determine the similarity between test cases. In these techniques, the goal is to give high priority to test cases that are vastly unalike (e.g., because they invoke different methods or have high string distances), thereby maximizing test case diversity and casting a wide net for detecting unique faults (Hemmati et al., 2011; Thomas et al., 2014).

However, using only string distance values, the possibility of having the same distance is quite high and may affect the prioritization process. Therefore, there was such an opportunity for improvement, as the prioritization was made based on the differences between two points; instead of using one metric only, which is distance, this string distances formulation may be enhanced more with another possible metric. Back to this sub-chapters objective, the challenges from this opening are, namely, how can string distances be enhanced with other metrics and, simultaneously, can give the necessary priority to test cases that are greatly altered. As supporting evidence, previous works reported prioritized test cases using string distance to have promising APFD values compared to randomly ordered ones (Bo Jiang & Chan, 2015; Ledru et al., 2012). However, the average APFD rank for almost all string distances was nearly identical (Ledru et al., 2012). A work using two string distances also faces similar problems (Wen et al., 2019). Hence, this implies an enhancement for string distances with other related metrics, such as the weighting scale, was worth further analysis.

As for the prioritization algorithm challenge, this research utilizes the generated test cases and prioritizes them based on their inputs to calculate the string distance/similarity between test cases. The fitness function that will be used is based on string similarity or distance between the test case and its neighboring test cases throughout the whole test suite. Assuming there are 1500 test cases, the number of permutations to complete evaluate the whole possible test cases sequences distances were 1500!, which is equal to 1500×1499×1498×…..×1 in value, and it is impossible for human manual prioritization. This situation is more likely similar to the traveling salesman problem (TSP), which is highly intractable (Bagloee & Asadi, 2015). The complexity arises from the size of test cases for the system. Thus, the worst-case running time for any algorithm for the TCP may increase exponentially with the number of test cases. For this reason, the challenges in the TCP algorithm for string metric can be recognized as; how prioritization algorithms should be tuned with sting metric, which could result in superior outcomes in terms of execution time and obtaining better APFD results for TCP itself.

Recent works (Hema Shankari et al., 2021; Bo Jiang & Chan, 2015; Panwar et al., 2018) show that the heuristic prioritization algorithm can significantly affect the TCP process. From their finding, using artificial intelligence algorithms may increase the average percentage of fault detection (APFD) results. However, existing prioritization often fails to provide the efficiency of coverage rate and execution time improvement. In this sense, a suitable prioritization algorithm that suits the challenges stated is worth investigating.

As time progressed, there were many new ML techniques introduced. These ML techniques can be categorized into several categories (Mahdieh et al., 2020; Malhotra, 2015). The categories can be named supervised, unsupervised, and reinforcement. The unsupervised ML technique is reserved when there is no historical information or incomplete information regarding the study program. The unsupervised ML technique may also have been chosen as it has been far less complex than the supervised ML technique (Hajri et al., 2019; Khalid & Qamar, 2019). The clustering technique was the most popular unsupervised ML technique in TCP. Work by Chen (Chen et al., 2018) proposed adaptive random sequences based on clustering techniques. By using black box information, their clustering techniques manage to cluster test cases as diverse as possible.

As the experiment is conducted further, the result shows that the technique manages to unfold fault at an earlier stage with higher effectiveness. Recent studies also show that the clustering technique may have high efficiency in terms of time execution, leading to cost-effectiveness (Harikarthik et al., 2019; Khalid & Qamar, 2019). However, some notable issues are outlined in the ML technique, specifically in clustering. One of them is the number of classes or clusters that could affect the results significantly (Chen et al., 2018). Apart from that, low performance on coverage rate due to imperfect differentiating the test cases do raise some attention in the TCP approach. Therefore, a well-tuned clustering ML technique with metric string enhancement could improve the coverage rate worth exploring. For this reason, the challenges for ML technique in TCP can recognize as; how ML technique can be tuned with the consideration of enhanced string metric algorithms to resolve the number of cluster issues which could result in superior outcomes in terms of coverage rate as well as obtaining better APFD result for TCP itself.

1.3 Research Gaps, Problem Statement and Research Questions

The study was conducted with the mean idea of systematically prioritizing test case prioritization (TCP) using the TCP string approach, driven by the problem arising from the string metrics formulations and its prioritization. Problems such as lack of metric consideration for the string metrics affect the uniqueness of each test case and may result in many similarities in the distance between test cases. In this sense, the prioritization algorithm often fails to offer an accurate path with desired distances. As for the ML technique, problems such as high numbers of clusters or classes formed due to differences in test cases resulted in a low coverage rate in TCP. In summary, the research gaps from the previous sub-chapter can be listed as follow:

i. String distance techniques have proven to be a technique that can work with any type of data but have a number of redundancy values that effected the average fault detection rate (APFD) and coverage rate (CR).

ii. Clustering techniques in TCP manage to cluster test cases based on changes but may have redundant priority values within the clusters, which reduces the APFD rate.

iii. TCP approaches that do not adequately address the process of String-Based TCP and Search-Based TCP.

Therefore, in this study, the TCP single string-based approach is addressed by the formulation of the weighted string distance metric, the prioritization algorithm for the dual factor string metric, and the clustering of the test case using the weighted string distance metric. Concretely, the macro research question of this study is:

"*How to effectively cluster and prioritize test cases using the weighted string distance approach for test case prioritization in software testing?*"

This research question is broad and requires various micro research questions to answer. The word effectiveness consists of several other metrics: fault detection rate, coverage rate, and execution time (Hao, Zhang, & Mei, 2016; Lou et al., 2019; Rothermel et al., 1999). To realize this research goal, four research questions need to be answered:

i. How to design the weighted string distances to ensure that string distances have a sufficient enhancement to improve APFD and CR in TCP?

ii. How to tune clustering and prioritization technique together with the weighted string distances to improve APFD rate in TCP?

iii. How to apply the proposed weighted string distance TCP approach with clustering technique into testing environment in order to improve the overall effectiveness of TCP?

1.4 Research Objectives

This study aims to establish a weighted string distance approach based on a modified clustering technique for optimizing test case prioritization. From the aim of the study and derived research questions, the following research objectives are defined, specifically:

i. To design a weighted string distance metric in test case prioritization by combining string distances and its weight-based metric to improve fault detection rate (APFD) and coverage rate (CR).

ii. To modify clustering technique and firefly algorithm with weighted string distance metric for optimizing test case prioritization to improve the APFD, CR, and timely execution results.

iii. To propose and evaluate the weighted string distance approach with clustering technique for optimizing test case prioritization on the benchmark datasets and its applicability to case study.

1.5 Scope of Study

The scopes of this research are limited within the following scopes:

i. The research focuses on small to medium-scale specialized systems available in many engineering applications.

ii. Dataset and benchmark programs with any type of change availability would be used to compare the findings of the enhanced test case prioritization technique to the existing test case prioritization.

iii. The research applied to software testing environments specifically on the test case prioritization domain only.

1.6 Thesis Structure and Organization

This thesis is outlined as follows:

Chapter 1 provides a brief overview of the research. It consists of a brief overview of software system development, software testing, test case prioritization techniques, and string metrics. Apart from that, this chapter highlight statement of the problem, research questions, aims and objectives of the study, and scope of the study.

Chapter 2 provides a brief overview of related works on test case prioritization. A summary of the literature review on TCP approaches is also presented. Besides that, string distances and ML and prioritization algorithms are briefly reviewed in the literature. Chapter 3 describes the overview of the research theoretical framework and research operational framework. This chapter also introduces application case studies and benchmark case studies used in this study for applicability and verification.

Chapter 4 elaborates on the implementation of four string distances: Manhattan, Levenshtein, Cosine Similarity, and Jaccard. Besides that, a proposed weighted string distance metric is implemented. Results are compared against the other four string distances. While in chapter 5 elaborate on the implementation of weighted string distance metric and prioritization with a heuristic algorithm applied, namely firefly. This serves as supporting results for the first objective and supporting evidence for further enhancement, leading to the implementation of the ML technique in the TCP approach. This chapter also elaborates on the method of implementing weighted string distance metric with the combination of the clustering technique. This serves as results for the second objective and supporting evidence for the effectiveness of the proposed double-string technique.

Chapter 6 elaborates on the proposed weighted string distance approach with the clustering technique for optimizing TCP. The methodology was then applied to the case study. The statistical evaluation of the result of the case study is also conducted in this chapter. To form a conclusion, Chapter 7 highlights the research achievement, research contribution, and future works.

REFERENCES

Abbad, A., & Tairi, H. (2016). Combining Jaccard and Mahalanobis Cosine distance to enhance the face recognition rate. *WSEAS Transactions on Signal Processing*, *16*, 171-178.

Abbeel, P., & Ng, A. Y. (2004, July). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning* (p. 1).

Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, *39*(1), 45-65.

Alsheikh, M. A., Lin, S., Niyato, D., & Tan, H. P. (2014). Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, *16*(4), 1996-2018.

Amato, G., Gennaro, C., & Savino, P. (2014). MI-File: using inverted files for scalable approximate similarity search. *Multimedia tools and applications*, *71*(3), 1333-1362.

Arafeen, M. J., & Do, H. (2013, March). Test case prioritization using requirements-based clustering. In *2013 IEEE sixth international conference on software testing, verification and validation* (pp. 312-321). IEEE.

Arora, P. K., & Bhatia, R. (2018). A systematic review of agent-based test case generation for regression testing. *Arabian Journal for Science and Engineering*, *43*(2), 447-470.

Azizi, M., & Do, H. (2018, April). A collaborative filtering recommender system for test case prioritization in web applications. In *Proceedings of the 33rd annual ACM symposium on applied computing* (pp. 1560-1567).

Bagherzadeh, M., Kahani, N., & Briand, L. (2021). Reinforcement learning for test case prioritization. *IEEE Transactions on Software Engineering*.

Bagloee, S. A., & Asadi, M. (2015). Prioritizing road extension projects with interdependent benefits under time constraint. *Transportation Research Part A: Policy and Practice*, *75*, 196-216.

Bajaj, A., & Sangwan, O. P. (2018, December). A survey on regression testing using nature-inspired approaches. In *2018 4th International Conference on Computing*

Communication and Automation (ICCCA) (pp. 1-5). IEEE.

Bajaj, A., & Sangwan, O. P. (2019). A systematic literature review of test case prioritization using genetic algorithms. *IEEE Access*, *7*, 126355-126375.

Carlson, R., Do, H., & Denton, A. (2011, September). A clustering approach to improving test case prioritization: An industrial case study. In *2011 27th IEEE International Conference on Software Maintenance (ICSM)* (pp. 382-391). IEEE.

Catal, C., & Mishra, D. (2013). Test case prioritization: a systematic mapping study. *Software Quality Journal*, *21*(3), 445-478.

Chaudhary, L., & Singh, B. (2019, January). Community detection using an enhanced louvain method in complex networks. In *International Conference on Distributed Computing and Internet Technology* (pp. 243-250). Springer, Cham.

Chen, J., Zhu, L., Chen, T. Y., Towey, D., Kuo, F. C., Huang, R., & Guo, Y. (2018). Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering. *Journal of Systems and Software*, *135*, 107-125.

Chittimalli, P. K., & Harrold, M. J. (2009). Recomputing coverage information to assist regression testing. *IEEE Transactions on Software Engineering*, *35*(4), 452-469.

Durelli, V. H., Durelli, R. S., Borges, S. S., Endo, A. T., Eler, M. M., Dias, D. R., & Guimaraes, M. P. (2019). Machine learning applied to software testing: A systematic mapping study. *IEEE Transactions on Reliability*, *68*(3), 1189-1212.

Eghbali, S., & Tahvildari, L. (2016). Test case prioritization using lexicographical ordering. *IEEE Transactions on software engineering*, *42*(12), 1178-1195.

Elbaum, S., Kallakuri, P., Malishevsky, A., Rothermel, G., & Kanduri, S. (2003). Understanding the effects of changes on the cost-effectiveness of regression testing techniques. *Software testing, verification and reliability*, *13*(2), 65-83.

Elbaum, S., Malishevsky, A. G., & Rothermel, G. (2002). Test case prioritization: A family of empirical studies. *IEEE transactions on software engineering*, *28*(2), 159-182.

Elbaum, S., Rothermel, G., Kanduri, S., & Malishevsky, A. G. (2004). Selecting a cost-effective test case prioritization technique. *Software Quality Journal*, *12*(3), 185-210.

Elbaum, S., Rothermel, G., & Penix, J. (2014, November). Techniques for improving regression testing in continuous integration development environments.

In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 235-245).

Fang, C., Chen, Z., Wu, K., & Zhao, Z. (2014). Similarity-based test case prioritization using ordered sequences of program entities. *Software Quality Journal*, *22*(2), 335-361.

Fisher, R. A. (1992). Statistical methods for research workers. In *Breakthroughs in statistics* (pp. 66-70). Springer, New York, NY.

Fu, W., Yu, H., Fan, G., & Ji, X. (2017). Coverage-based clustering and scheduling approach for test case prioritization. *IEICE TRANSACTIONS on Information and Systems*, *100*(6), 1218-1230.

Furcy, D., & Koenig, S. (2005). Limited discrepancy beam search. In *Proceedings of the 19th international joint conference on Artificial intelligence* (pp. 125-131).

Garousi, V., Özkan, R., & Betin-Can, A. (2018). Multi-objective regression test selection in practice: An empirical study in the defense software industry. *Information and Software Technology*, *103*, 40-54.

Gokilavani, N., & Bharathi, B. (2021). Test case prioritization to examine software for fault detection using PCA extraction and K-means clustering with ranking. *Soft Computing*, *25*(7), 5163-5172.

Gokilavani, N., & Bharathi, B. (2021). Multi-Objective based test case selection and prioritization for distributed cloud environment. *Microprocessors and Microsystems*, *82*, 103964.

Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *international journal of Computer Applications*, *68*(13), 13-18.

González-Estrada, E., & Cosmes, W. (2019). Shapiro–Wilk test for skew normal distributions based on data transformations. *Journal of Statistical Computation and Simulation*, *89*(17), 3258-3272.

Hajri, I., Goknil, A., Pastore, F., & Briand, L. C. (2020). Automating system test case classification and prioritization for use case-driven testing in product lines. *Empirical Software Engineering*, *25*(5), 3711-3769.

Hao, D., Zhang, L., & Mei, H. (2016). Test-case prioritization: achievements and challenges. *Frontiers of Computer Science*, *10*(5), 769-777.

Hao, D., Zhang, L., Zang, L., Wang, Y., Wu, X., & Xie, T. (2015). To be optimal or not in test-case prioritization. *IEEE Transactions on Software Engineering*, *42*(5), 490-505.

Hao, D., Zhang, L., Zhang, L., Rothermel, G., & Mei, H. (2014). A unified test case prioritization approach. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *24*(2), 1-31.

Haraty, R. A., Mansour, N., Moukahal, L., & Khalil, I. (2016). Regression test cases prioritization using clustering and code change relevance. *International Journal of Software Engineering and Knowledge Engineering*, *26*(05), 733-768.

Harikarthik, S. K., Palanisamy, V., & Ramanathan, P. (2019). Optimal test suite selection in regression testing with testcase prioritization using modified Ann and Whale optimization algorithm. *Cluster Computing*, *22*(5), 11425-11434.

Harikarthik, S. K., Ramanathan, P., & Palanisamy, V. (2018). 'Enhancement of regression testing using genetic data generation and test case prioritization using m-ACO technique. *International Journal of Engineering & Technology*, *7*(1.3), 95-99.

Hasnain, M., Ghani, I., Pasha, M. F., Malik, I. H., & Malik, S. (2019). Investigating the Regression Analysis Results for Classification in Test Case Prioritization: A Replicated Study. *International Journal of Internet, Broadcasting and Communication*, *11*(2), 1-10.

Hema Shankari, K., Mathivilasini, S., Arasu, D., & Suseendran, G. (2021). Genetic Algorithm Based on Test Suite Prioritization for Software Testing in Neural Network. In *Proceedings of First International Conference on Mathematical Modeling and Computational Science* (pp. 409-416). Springer, Singapore.

Hemmati, H., Arcuri, A., & Briand, L. (2011, March). Empirical investigation of the effects of test suite properties on similarity-based test case selection. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation* (pp. 327-336). IEEE.

Hemmati, H., Arcuri, A., & Briand, L. (2010, November). Reducing the cost of model-based testing through test case diversity. In *IFIP International Conference on Testing Software and Systems* (pp. 63-78). Springer, Berlin, Heidelberg.

Hettiarachchi, C., Do, H., & Choi, B. (2016). Risk-based test case prioritization using a fuzzy expert system. *Information and Software Technology*, *69*, 1-15.

Huang, A. (2008, April). Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand* (Vol. 4, pp. 9-56).

Huang, Y. C., Peng, K. L., & Huang, C. Y. (2012). A history-based cost-cognizant test case prioritization technique in regression testing. *Journal of Systems and Software*, *85*(3), 626-637.

Jiang, B., & Chan, W. K. (2015). Input-based adaptive randomized test case prioritization: A local beam search approach. *Journal of Systems and Software*, *105*, 91-106.

Jimenez, S., Gonzalez, F. A., & Gelbukh, A. (2016). Mathematical properties of soft cardinality: Enhancing Jaccard, Dice and cosine similarity measures with element-wise distance. *Information Sciences*, *367*, 373-389.

Kazmi, R., Jawawi, D. N., Mohamad, R., & Ghani, I. (2017). Effective regression test case selection: A systematic literature review. *ACM Computing Surveys (CSUR)*, *50*(2), 1-32.

Kanimozhi, R., & Rbalakrishnan, J. (2014). Cosine similarity based clustering for software testing using prioritization. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *16*(1), 75-80.

Kennedy, J. (2011). Particle swarm optimization. Encyclopedia of machine learning. *Springer*, *760*, 766.

Khalid, Z., & Qamar, U. (2019, October). Weight and cluster based test case prioritization technique. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 1013-1022). IEEE.

Khalilian, A., Azgomi, M. A., & Fazlalizadeh, Y. (2012). An improved method for test case prioritization by incorporating historical test case data. *Science of Computer Programming*, *78*(1), 93-116.

Kim, J. M., & Porter, A. (2002, May). A history-based test prioritization technique for regression testing in resource constrained environments. In *Proceedings of the 24th international conference on software engineering* (pp. 119-129).

Konsaard, P., & Ramingwong, L. (2015, June). Total coverage based regression test case prioritization using genetic algorithm. In *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)* (pp. 1-6). IEEE.

Kumar, A., Vembu, S., Menon, A. K., & Elkan, C. (2013). Beam search algorithms for multilabel learning. *Machine learning*, *92*(1), 65-89.

Kumar, A., & Singh, K. (2014). A Literature Survey on test case prioritization. *Compusoft*, *3*(5), 793-799.

L Lachmann, R. (2018, June). Machine learning-driven test case prioritization approaches for black-box software testing. In *The European test and telemetry conference, Nuremberg, Germany*.

Lachmann, R., Schulze, S., Nieke, M., Seidl, C., & Schaefer, I. (2016, December). System-level test case prioritization using machine learning. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 361-368). IEEE.

Lancia, G., & Dalpasso, M. (2019, August). Speeding-Up the Dynamic Programming Procedure for the Edit Distance of Two Strings. In *International Conference on Database and Expert Systems Applications* (pp. 59-66). Springer, Cham.

Ledru, Y., Petrenko, A., Boroday, S., & Mandran, N. (2012). Prioritizing test cases with string distances. *Automated Software Engineering*, *19*(1), 65-95.

Li, N., Francis, P., & Robinson, B. (2008, November). Static detection of redundant test cases: An initial study. In *2008 19th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 303-304). IEEE.

Lou, Y., Chen, J., Zhang, L., & Hao, D. (2019). A survey on regression test-case prioritization. In *Advances in Computers* (Vol. 113, pp. 1-46). Elsevier.

Lousada, J., & Ribeiro, M. (2020). Reinforcement learning for test case prioritization. *arXiv preprint arXiv:2012.11364*.

Luo, Q., Moran, K., & Poshyvanyk, D. (2016, November). A large-scale empirical comparison of static and dynamic test case prioritization techniques. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 559-570).

Ma, T., Zeng, H., & Wang, X. (2016, May). Test case prioritization based on requirement correlations. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)* (pp. 419-424). IEEE Computer Society.

Mahdieh, M., Mirian-Hosseinabadi, S. H., Etemadi, K., Nosrati, A., & Jalali, S. (2020). Incorporating fault-proneness estimations into coverage-based test case prioritization methods. *Information and Software Technology*, *121*, 106269.

Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, *27*, 504-518.

Marijan, D., Gotlieb, A., & Sen, S. (2013, September). Test case prioritization for continuous regression testing: An industrial case study. In *2013 IEEE*

International Conference on Software Maintenance (pp. 540-543). IEEE.

McRoberts, R. E., Næsset, E., & Gobakken, T. (2015). Optimizing the k-Nearest Neighbors technique for estimating forest aboveground biomass using airborne laser scanning data. *Remote Sensing of Environment*, *163*, 13-22.

Mece, E. K., Paci, H., & Binjaku, K. (2020). The application of machine learning in test case prioritization-a review. *European Journal of Electrical Engineering and Computer Science*, *4*(1).

Mei, L., Chan, W. K., Tse, T. H., Jiang, B., & Zhai, K. (2014). Preemptive regression testingof workflow-based web services. *IEEE Transactions on Services Computing*, *8*(5), 740-754.

Mete, S., Çil, Z. A., Ağpak, K., Özceylan, E., & Dolgui, A. (2016). A solution approach based on beam search algorithm for disassembly line balancing problem. *Journal of Manufacturing Systems*, *41*, 188-200.

Miranda, B., & Bertolino, A. (2017). Scope-aided test prioritization, selection and minimization for software reuse. *Journal of Systems and Software*, *131*, 528-549.

Miranda, B., Cruciani, E., Verdecchia, R., & Bertolino, A. (2018, May). FAST approaches to scalable similarity-based test case prioritization. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)* (pp. 222-232). IEEE.

Mohd-Shafie, M. L., Kadir, W. M. N. W., Lichter, H., Khatibsyarbini, M., & Isa, M. A. (2021). Model-based test case generation and prioritization: a systematic literature review. *Software and Systems Modeling*, 1-37.

Mohd-Shafie, M. L., Wan-Kadir, W. M. N., Khatibsyarbini, M., & Isa, M. A. (2020). Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion. *PloS one*, *15*(2), e0229312.

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.

Muthusamy, T., & Seetharaman, K. (2014). A new effective test case prioritization for regression testing based on prioritization algorithm. *Int. J. Appl. Inf. Syst.(IJAIS)*, *6*(7), 21-26.

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Nagar, R., Kumar, A., Singh, G. P., & Kumar, S. (2015, February). Test case selection and prioritization using cuckoos search algorithm. In *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)* (pp. 283-288). IEEE.

Nguyen, A., Le, B., & Nguyen, V. (2019, September). Prioritizing automated user interface tests using reinforcement learning. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering* (pp. 56-65).

Nurmuradov, D., Bryce, R., Piparia, S., & Bryant, B. (2018). Clustering and combinatorial methods for test suite prioritization of GUI and web applications. In *Information Technology-New Generations* (pp. 459-466). Springer, Cham.

Pan, R., Bagherzadeh, M., Ghaleb, T. A., & Briand, L. (2022). Test case selection and prioritization using machine learning: a systematic literature review. *Empirical Software Engineering*, *27*(2), 1-43.

Panwar, D., Tomar, P., Harsh, H., & Siddique, M. H. (2018). Improved meta-heuristic technique for test case prioritization. In *Soft computing: Theories and applications* (pp. 647-664). Springer, Singapore.

Prado Lima, J. A. P., & Vergilio, S. R. (2020). Test Case Prioritization in Continuous Integration environments: A systematic mapping study. *Information and Software Technology*, *121*, 106268.

Prado Lima, J. A., & Vergilio, S. R. (2020). A multi-armed bandit approach for test case prioritization in continuous integration environments. *IEEE Transactions on Software Engineering*.

Radovanović, M., Nanopoulos, A., & Ivanović, M. (2014). Reverse nearest neighbors in unsupervised distance-based outlier detection. *IEEE transactions on knowledge and data engineering*, *27*(5), 1369-1382.

Rattan, P., Arora, M., Rakhra, M., & Goel, V. (2021). A Neoteric Approach of Prioritizing Regression Test Suites Using Hybrid ESDG Models. *Annals of the Romanian Society for Cell Biology*, 2965-2973.

Rezvani, M., & Hashemi, S. M. (2012, December). Enhancing accuracy of topic sensitive PageRank using jaccard index and cosine similarity. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology* (Vol. 1, pp. 620-624). IEEE.

Rosenbauer, L., Stein, A., Pätzel, D., & Hähner, J. (2020, December). XCSF with experience replay for automatic test case prioritization. In *2020 IEEE Symposium*

Series on Computational Intelligence (SSCI) (pp. 1307-1314). IEEE.

Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (1999, August). Test case prioritization: An empirical study. In *Proceedings IEEE International Conference on Software Maintenance-1999 (ICSM'99).'Software Maintenance for Business Change'(Cat. No. 99CB36360)* (pp. 179-188). IEEE.

Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (2001). Prioritizing test cases for regression testing. *IEEE Transactions on software engineering*, *27*(10), 929-948.

Saini, M., Sharma, D., & Gupta, P. K. (2011, November). Enhancing information retrieval efficiency using semantic-based-combined-similarity-measure. In *2011 International Conference on Image Information Processing* (pp. 1-4). IEEE.

Sampath, S., Bryce, R., & Memon, A. M. (2013). A uniform representation of hybrid criteria for regression testing. *IEEE transactions on software engineering*, *39*(10), 1326-1344.

Sánchez, A. B., Segura, S., & Ruiz-Cortés, A. (2014, March). A comparison of test case prioritization criteria for software product lines. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation* (pp. 41-50). IEEE.

Shahbazi, A., & Miller, J. (2015). Black-box string test case generation through a multi-objective optimization. *IEEE Transactions on Software Engineering*, *42*(4), 361-378.

Singh, Y., Kaur, A., Suri, B., & Singhal, S. (2012). Systematic literature review on regression test prioritization techniques. *Informatica*, *36*(4).

Spieker, H., Gotlieb, A., Marijan, D., & Mossige, M. (2017, July). Reinforcement learning for automatic test case prioritization and selection in continuous integration. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 12-22).

Srikanth, H., Hettiarachchi, C., & Do, H. (2016). Requirements based test prioritization using risk factors: An industrial study. *Information and Software Technology*, *69*, 71-83.

Stallbaum, H., Metzger, A., & Pohl, K. (2008, May). An automated technique for risk-based test case generation and prioritization. In *Proceedings of the 3rd international workshop on Automation of software test* (pp. 67-70).

Stratis, P., & Rajan, A. (2016, August). Test case permutation to improve execution time. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering* (pp. 45-50).

Su, W., Li, Z., Wang, Z., & Yang, D. (2020, March). A meta-heuristic test case prioritization method based on hybrid model. In *2020 International Conference on Computer Engineering and Application (ICCEA)* (pp. 430-435). IEEE.

Tahvili, S., Afzal, W., Saadatmand, M., Bohlin, M., Sundmark, D., & Larsson, S. (2016). Towards earlier fault detection by value-driven prioritization of test cases using fuzzy TOPSIS. In *Information Technology: New Generations* (pp. 745-759). Springer, Cham.

Tahvili, S., Hatvani, L., Felderer, M., Afzal, W., & Bohlin, M. (2019, April). Automated functional dependency detection between test cases using doc2vec and clustering. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)* (pp. 19-26). IEEE.

Thomas, S. W., Hemmati, H., Hassan, A. E., & Blostein, D. (2014). Static test case prioritization using topic models. *Empirical Software Engineering*, *19*(1), 182-212.

Unkovskiy, A., Bui, P. H. B., Schille, C., Geis-Gerstorfer, J., Huettig, F., & Spintzyk, S. (2018). Objects build orientation, positioning, and curing influence dimensional accuracy and flexural properties of stereolithographically printed resin. *Dental Materials*, *34*(12), e324-e333.

Wang, X., & Zeng, H. (2016). History-based dynamic test case prioritization for requirement properties in regression testing. In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery* (pp. 41-47).

Wang, Y., Zhao, X., & Ding, X. (2015, September). An effective test case prioritization method based on fault severity. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)* (pp. 737-741). IEEE.

Wen, Z., Deng, D., Zhang, R., & Kotagiri, R. (2019, April). : An efficient entity extraction algorithm using two-level edit-distance. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (pp. 998-1009). IEEE.

Wen, Z., Deng, D., Zhang, R., & Ramamohanarao, K. (2017). A technical report: entity extraction using both character-based and token-based similarity. *arXiv preprint arXiv:1702.03519*.

Wu, Z., Yang, Y., Li, Z., & Zhao, R. (2019, October). A time window based reinforcement learning reward for test case prioritization in continuous integration. In *Proceedings of the 11th Asia-Pacific Symposium on Internetware* (pp. 1-6).

Xia, X., Gong, L., Le, T. D. B., Lo, D., Jiang, L., & Zhang, H. (2016). Diversity maximization speedup for localizing faults in single-fault and multi-fault programs. *Automated Software Engineering*, *23*(1), 43-75.

Xiao, C., Wang, W., & Lin, X. (2008). Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *Proceedings of the VLDB Endowment*, *1*(1), 933-944.

Xiao, L., Miao, H., & Zhong, Y. (2018). Test case prioritization and selection technique in continuous integration development environments: a case study. *International Journal of Engineering & Technology*, *7*(2.28), 332-336.

Xiao, L., Miao, H., Zhuang, W., & Chen, S. (2017, May). An empirical study on clustering approach combining fault prediction for test case prioritization. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)* (pp. 815-820). IEEE.

Xiao, L., Miao, H., Zhuang, W., & Chen, S. (2016, November). Applying Assemble Clustering Algorithm and Fault Prediction to Test Case Prioritization. In *2016 International Conference on Software Analysis, Testing and Evolution (SATE)* (pp. 108-116). IEEE.

Yadav, D. K., & Dutta, S. (2016). Test case prioritization technique based on early fault detection using fuzzy logic. In *2016 3rd international conference on computing for sustainable global development (INDIACom)* (pp. 1033-1036).

Yadav, D. K., & Dutta, S. (2017). Regression test case prioritization technique using genetic algorithm. In *Advances in computational intelligence* (pp. 133-140). Springer, Singapore.

Yan, M., Xia, X., Lo, D., Hassan, A. E., & Li, S. (2019). Characterizing and identifying reverted commits. *Empirical Software Engineering*, *24*(4), 2171-2208.

Yang, Y., Li, Z., Shang, Y., & Li, Q. (2021). Sparse reward for reinforcement learning-based continuous integration testing. *Journal of Software: Evolution and Process*, e2409.

Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: a survey. *Software testing, verification and reliability*, *22*(2), 67-120.

Yoon, M., Lee, E., Song, M., & Choi, B. (2012). A test case prioritization through correlation of requirement and risk. *Journal of Software Engineering and Applications*, *5*(10), 823.

Yu, Y. T., & Lau, M. F. (2012). Fault-based test suite prioritization for specification-based testing. *Information and Software Technology*, *54*(2), 179-202.

Zhang, K., Zhang, Y., Zhang, L., Gao, H., Yan, R., & Yan, J. (2020). Neuron activation frequency based test case prioritization. In *2020 International Symposium on Theoretical Aspects of Software Engineering (TASE)* (pp. 81-88).

Zhang, T., Wang, X., Wei, D., & Fang, J. (2018). Test case prioritization technique based on error probability and severity of UML models. *International Journal of Software Engineering and Knowledge Engineering*, *28*(06), 831-844.

Zhang, W., Qi, Y., Zhang, X., Wei, B., Zhang, M., & Dou, Z. (2019, August). On test case prioritization using ant colony optimization algorithm. In *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)* (pp. 2767-2773). IEEE.

Zhang, X. Y., Towey, D., Chen, T. Y., Zheng, Z., & Cai, K. Y. (2016, May). A random and coverage-based approach for fault localization prioritization. In *2016 Chinese Control and Decision Conference (CCDC)* (pp. 3354-3361).

Zhao, X., Wang, Z., Fan, X., & Wang, Z. (2015). A clustering-bayesian network based approach for test case prioritization. In *2015 IEEE 39th Annual Computer Software and Applications Conference* (Vol. 3, pp. 542-547). IEEE.

Zhou, Z. Q., Sinaga, A., & Susilo, W. (2012). On the fault-detection capabilities of adaptive random test case prioritization: Case studies with large test suites. In *2012 45th Hawaii International Conference on System Sciences* (pp. 5584-5593). IEEE.

LIST OF PUBLICATIONS

Journal with Impact Factor

1. **Khatibsyarbini**, M., Isa, M. A., Jawawi, D. N., Hamed, H. N. A., & Suffian, M. D. M. (2019). Test case prioritization using firefly algorithm for software testing. *IEEE access*, *7*, 132360-132373. **(Q2, IF:1.01)**
2. **Khatibsyarbini**, M., Isa, M. A., Jawawi, D. N., Shafie, M. L. M., Kadir, W. M. N. W., Hamed, H. N. A., & Suffian, M. D. M. (2021). Trend Application of Machine Learning in Test Case Prioritization: A Review on Techniques. *IEEE Access*. **(Q2, IF:1.01)**