A STUDY OF TEST CASE PRIORITIZATION TECHNIQUE BASED ON STRING DISTANCE METRICS

MUHAMMAD KHATIBSYARBINI

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

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

.

JANUARY 2019

Dedicated to:

My Beloved Parent

My Lovely Wife

My Righteous Son

My Respected Lecturers

My Dear Brothers

Thank you for your prayers and supports

ACKNOWLEDGEMENT

Praise to Allah swt for His blessings that I am able to finish this research. Firstly, I would like to express my gratitude to my parents and my wife for supporting me spiritually throughout writing this thesis and my life in general.

Secondly, I would like to express my sincere gratitude to my advisor Dr. Mohd Adham Isa for the continuous support of my Master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master study.

Besides my advisor, I would like to thank Associate Prof. Dr. Dayang Norhayati Abang Jawawi and Associate Prof. Dr. Wan Mohd Nasir Wan Kadir, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Last but not the least, I would like to thank my friends who helped and supporting me in any aspect throughout this research.

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). There are different approaches and the process for each approach varies. Furthermore, these approaches are not well documented within the single TCP approach. Based on current studies, having an approach that has high coverage effectiveness (CE) and APFD rate, remains a challenge in TCP. The string-based approach is known to have a single string distance based metric to differentiate test cases that can improve the CE results. However, to differentiate precisely the test cases, the string distances require enhancement. Therefore, a TCP technique based on string distance metric was developed to improve CE and APFD rate. In this research, to differentiate precisely the test cases and counter the string distances problem, an enhanced string distances based metric with a string weight based metric was introduced. Then, the metric was executed under designed process for string-based approach for complete evaluation. Experimental results showed that the enhanced string metric had the highest APFD with 98.56% and highest CE with 69.82% in Siemen dataset, *cstcas*. Besides, the technique yielded the highest APFD with 76.38% in Robotic Wheelchair System (RWS) case study. As a conclusion, the enhanced TCP technique with weight based metric has prioritised the test case based on their occurrences which helped to differentiate precisely the test cases, and improved the overall scores of APFD and CE.

ABSTRAK

Banyak pendekatan keutamaan ujian (TCP) telah diperkenalkan untuk meningkatkan daya maju ujian dalam aktiviti ujian perisian dengan matlamat untuk memaksimumkan peratusan purata peratusan kesalahan awal (APFD). Terdapat banyak perbezaan dalam proses untuk setiap pendekatan yang ada. Tambahan pula, pendekatan-pendekatan ini tidak didokumenkan dengan lengkap dalam setiap TCP proses. Berdasarkan kajian semasa, untuk mempunyai pendekatan yang mempunyai keberkesanan liputan(CE) dan kadar APFD yang tinggi, masih menjadi cabaran dalam TCP. Pendekatan berasaskan rentetan telah menunjukkan bahawa dengan menggunakan metrik jarak tunggal untuk membezakan kes ujian dapat meningkatkan hasil CE. Walau bagaimanapun, untuk membezakan kes ujian dengan tepat, jarak rentetan masih memerlukan peningkatan. Oleh itu, satu teknik pengutamaan kes ujian berdasarkan jarak jarak metrik telah dibangunkan untuk meningkatkan kadar hasil CE dan APFD. Dalam kajian ini, untuk mengatasi masalah jarak rentetan dan mengira jarak rentetan dengan tepat, metrik berasaskan jarak rentetan digabungkan dengan metrik berasaskan berat rentetan. Kemudian, metrik gabungan ini dilaksanakan di bawah proses yang direka untuk pendekatan berasaskan rentak untuk penilaian lengkap. Hasil percubaan menunjukkan metrik gabungan ini mempunyai kadar APFD tertinggi dengan 98.56% dan CE tertinggi dengan 69.82% dalam kumpulan data Siemen iaitu *cstcas*. Selain itu, teknik hasil gabungan metrik ini mendapat kadar APFD yang lebih tinggi dengan 76.38% dalam kajian kes Sistem Robot Kerusi Roda (RWS). Sebagai kesimpulan, teknik yang dibangunkan telah memberi keutamaan berbeza kepada setiap kes ujian yang mana telah membantu dalam membezakan setiap kes, sekali gus meningkatkan skor keseluruhan APFD dan CE.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

AI	-	Artificial Intelligent
APFD	-	Average Percentage Fault Detection
ANOVA	-	Analysis of Variance
CE	-	Coverage Effectiveness
CS	-	Cosine Similarity
FATE	-	Fault Adequate Test Size
GA	-	Genetic Algorithm
HSD	-	High Significant Different
JC	-	Jaccard
L	-	Levenshtein
LOC	-	Line of Code
M	-	Manhattan
RWS	-	Robotic Wheelchair System
SLR	-	Systematic Literature Review
TCP	-	Test Case Prioritization
TF-IDF	-	Term Frequency – Inverse Document Frequency
TSP	-	Travelling Salesman Problem

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

Software engineering is not only confined to programming and software development efforts (van Katwijk, 1991). Software engineering itself is an implementation of engineering procedures in the development of any specific software in a much systematic way. Within software development process, software testing consumes a significant amount of time and can be the most expensive phase (Myers *et al*., 2004). Software testing is arguably the least understood part of a software development process. Software testing itself involves iterative strategies, which are often subjected to various pressures due to time constraint and fixed resources. Software engineering communities are regularly compelled to pre-maturely end their testing activities, attributed to financial stress and time necessities, which could lead to the generation of various conflicts relating to software quality and client agreement.

In practical sense, developers are aware of the frustration arising from software bugs that are reported by users. When this happens, developers inevitably ask: How did these bugs escape watchful eyes in testing? Countless hours went into a series of meticulous testing of hundreds or thousands of variables and code statements, so how could a bug have eluded such vigilance? The answer might lie within the software testing activity itself. Did testers test all possible test cases? Were all possible ordering of statements tested? An immediate solution is to run all test cases using several testing strategies, which may help testers to reveal the drawbacks of each strategy, such as time execution and effectiveness of fault detection. In light of this, it has been reported that the application of test case prioritization (TCP) appears to enhance test viability in software testing activity (Rothermel *et al*., 1999).

TCP approach was first mentioned in the work of Wong *et al*. (1997). That work, however, only applied prioritization on test cases that had undergone test case selection. Later, Rothermel and Harold proposed and evaluated the TCP approach in a much broader context. Consider a test suite as listed in Table 1.1 (Elbaum *et al*., 2000). This example only depicts an ideal situation in which fault detection information is known.

Table 1.1 Test suite example

Test Case	Fault revealed by test case									
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
TC1	√			√	√			√	√	√
TC2	√				√	√	√			
TC3					√	√	√	√	√	√
TC4	√	√	√		√			√	√	
TC5	√	√	√	√	√					

TCP sort test cases with the highest significance first according to some measures. Consequently, the primary goal of prioritization is to maximize early fault detection. Referring to Table 1.1, it can be concluded that the ordering of test cases in the order of TC5-TC3 is a much superior ordering that any other combinations. Such ordering detects all of the faults at an earlier rate. In practice, it is often challenging to distinguish which tests will essentially will reveal faults. Hence, the effectiveness of a test case prioritization largely depends on choosing the most applicable approach from a pool of approaches, expecting that an early intensification of a certain approach will result in yielding an earlier fault discovery. There are many dimensions of test case prioritization approaches. Eight broad dimensions were described by Singh (2012). For each approach, the scholar specified potential values, advantages, and limitation.

In software engineering research, inputs and dataset types play important roles that allow scholars to determine their advantages and limitations. As there are various approaches that exist, processes that are involved may be specific to each approach. Variation of processes that is unique to each approach benefits project managers, as they are able to adapt suitable approach that fits project schedules, in order to compensate constraints that exist within the project development process. Despite TCP being a relatively mature approach, there is a dearth of available documentations that describe a systematic process within single TCP approach. In view of this, there is a gap of readily accessible systematic process that facilitates a complete TCP within existing approaches, which often involve the utilization of distinctive resources and processes.

In a scenario where the only accessible resources are test cases and code changes from previous working system, TCP approach may be used. Particularly, TCP approach that utilizes string metric, as this approach is capable of distinguishing the differences in test cases followed by prioritization based on string similarity. In recent years, numerous TCP works have been documented, which prioritize test cases solely based on information related to test cases (Bo Jiang and Chan, 2015; Ledru *et al*., 2012; Mei, Cai, *et al*., 2015; Thomas *et al*., 2014). By depending on information that is available from test cases such as test case inputs, software tester may prioritize test cases prior to the availability of system source code. Such strategy reduces the time spent to prioritize test cases as complete source code is only available at a much later development phase.

String distance, which computes textual similarities, could be used to differentiate test cases. This allows prioritization to be executed as test cases are subsequently assigned distance or weight. String metrics play an important role in textual document-related research 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 (Gomaa and Fahmy, 2013). String metrics can be categorized based on their metric calculation such as distances, similarity and weight. However, to precisely calculate the distance between test cases, a specific and reliable string distance with specific priority is required. In existing works (Bo Jiang and Chan, 2015; Ledru *et al*., 2012), only single-based string distance was used, which may yield redundancy in equivalence distance. In order to overcome this, enhancement of string distances may be pursued.

Subsequent step upon the calculation of string distance is the application of prioritization algorithms to prioritize test cases based on their respective string distance values. Recent work by Bo Jiang and Chan (2015) demonstrates that heuristic prioritization algorithm can give a significant effect to TCP process. From their findings, the application of artificial intelligence algorithm may increase the results of average percentage of fault detection (APFD). However, existing prioritization approaches with string metric provide less favorable coverage effectiveness and execution time performance.

The main challenge to the problems alluded can be divided into two primary issues, namely: systematic process for string distance technique in TCP and string distance formulation. String distances are essentially formulations used to determine textual distances to morph a test case to another test case. This is achieved by; either calculating the difference, or similarity of test cases based on their attributes such as test case inputs. As for the process, it is meant to provide a systematic guidance on how to execute TCP process with consideration of string distances. These challenges aim to address the issues of systematic maximization of fault detection in test case prioritization, whereby, specific problems will be explained in detail in the following sections.

1.2 Challenges in String Distance based TCP

In software development life cycle, product being maintained is often subjected to system changes. After every change is implemented, immediate testing is required to ensure that the software adheres or meets specification. Assuming software testing team is required to execute testing and the only available resource is test suite with related attributes, TCP process needs to work out a strategy that prioritizes the test cases using available information. Work by Bo Jiang and Chan (2015) attempts to maximize test case diversity through test case input information, which differs from the work of Ledru *et al*. (2012). In Ledru *et al*. (2012), each test case is treated as a string of characters, and prioritization of test cases is carried out 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 (i.e., because they invoke different methods, or have higher string distance values), thereby maximizing test case diversity and casting a wide net for detecting unique faults (Hemmati *et al*., 2011).

However, by relying solely on string distance values, the possibility of obtaining equal distances among test cases is relatively high and may affect overall prioritization process. Associated with this issue, there is much room for improvement to be made, as prioritization is primarily based on the differences between two points. Instinctively, instead of using a single string distance, the formulation may be enhanced further via combination with other possible string distances. Primary challenge that arises from this notion is: How can string distances be enhanced with other metrics while at the same time provide necessary priority weights to test cases that are greatly altered? As supporting evidence, previous works reported that prioritized test cases using string distance have promising APFD values as compared to randomly ordered test cases (Bo Jiang and Chan, 2015; Ledru *et al*., 2012). Despite this, average scores of APFD ranks across almost all string distances are nearly identical, as reported in the work of Ledru *et al*. (2012). Hence, this implies that an enhancement of string distances with other related metrics such as weighting scale is worth further analysis.

1.3 Challenge in Process for TCP String-Based Technique

Software engineering highly concerns on how the engineering processes are applied into software development in a systematic way. Therefore, it is necessary to have a systematic process for TCP approach, particularly for string-based TCP approach. There are numerous works that exhibit highly identical process flows, with the only notable difference lies; either in the addition, or the reduction of one step to an existing process flow (Bo Jiang and Chan, 2015; Ledru *et al*., 2012; Shahbazi and Miller, 2016). Variation of process flows may yield different results despite of utilization of a similar TCP approach on identical datasets. Therefore, the challenge in this process can be highlighted as: How to apply a string-based TCP technique into a testing environment in order to improve the effectiveness of the process?

Generally, a TCP process begins with the preparation of data. Even though the description of this step is almost non-existent in existing literature, it is compulsory for any experiment or research endeavor to identify which information or data that shall be used. 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. The process proceeds with prioritizing the calculated criteria or dependency. Finally, the performance is measured. This advocates the needs of formally defined steps and process, centered on string-based approach, with the challenges that are worth to be addressed.

1.4 Research Questions

The study of TCP approaches produces several research gaps worth exploring. There are numerous approaches that have been adapted in the field of TCP which concern with system evolution. Even though most existing works tend to merely focus on TCP approaches, several other works cover the processes that are required to apply proposed TCP approaches. As for string-based TCP approach, highly redundant test cases owning identical string distances lead to a lack of accuracy and efficiency along the prioritization process, especially in terms of APFD scores. These problems could further lead to an un-systematic and inaccurate TCP. Therefore, it is a primary focus of this research to develop a systematic testing process for a string-based TCP approach. Consequently, a macro research question of this research is:

"*How to increase test case prioritization effectiveness with string distance systematically?*"

The macro research question leads to several micro research questions. 'Effectiveness' itself could be quantified based on several measurements including fault detection rate, coverage effectiveness, and execution time. There are two micro research questions that need to be answered:

i. What should be combined to the string distances to ensure that string distances have sufficient enhancement to increase fault detection rate?
ii. How to apply the proposed technique systematically into testing environment to improve the effectiveness of the process?

1.5 Research Objectives

The goal of this study is to establish a preliminary testing involving a test case prioritization approach to adapt the changes in the source code of a system. From the aim of the study and derived research questions, the following research objectives are defined, specifically:

i. To propose an enhanced string metric in test case prioritization by combining string distances and its weight-based metric to increase fault detection rate.
ii. To propose a process for string-based TCP approach to evaluate the effectiveness of the proposed TCP process on benchmark programs and its applicability on case studies systematically.

1.6 Scope of Study

The scopes of this research are limited to the following:

i. The research focuses on small- to medium-scale specialized systems which are available in many engineering applications.
ii. Benchmark programs and a case study would be used to compare the findings of the enhanced test case prioritization approach to existing test case prioritization approaches.

1.7 Significances and Original Contributions of Study

The research on test case prioritization technique is important in the context of safety-critical embedded system as it can contribute to uplifting a system's software testing process. Moreover, the research conducted contributes to a better testing quality. Through this research:

i. Prioritization of test cases can be performed at a much-reduced time.
ii. Safety criterion which is an important factor for safety-critical systems is enforced as an important prioritization element when systems undergo any changes.

iii. Fault detection capability of the proposed test case prioritization approach is significantly increased.

iv. The proposed approach exhibits statistical significance.

1.8 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, within this chapter, statement of the problem, motivation of study, aims of study, objectives of the study, justification of study, scope of study, and the significance of the study are elaborated as well.

Chapter 2 provides brief overviews of related works on test case prioritization. A summary of systematic literature review on test case prioritization approaches is also presented. Besides that, string distances and prioritization algorithm are briefly reviewed in this chapter.

Chapter 3 describes the overview of the research theoretical framework and research operational framework. This chapter also introduces case studies and benchmark programs, which will be utilized in later chapters for applicability and verification.

Chapter 4 elaborates the implementation of four string distances namely, Manhattan, Levenshtein, Cosine Similarity and Jaccard. Besides that, a proposed enhanced string distance is implemented. Results are compared against the other four string distances.

Chapter 5 elaborates the proposed process for string-based test case prioritization. The process is then applied to one case study. Statistical evaluation of the case study is also conducted in this chapter.

Chapter 6 provides the conclusion, contribution, limitations, and future works of this research.

References

[1] P. Ralph, "Software engineering process theory: A multi-method comparison of Sensemaking-Coevolution-Implementation Theory and function-behavior-structure theory," *Information and Software Technology*, vol. 70, pp. 232–250, 2016.

[2] G. J. Myers, T. M. Thomas, and C. Sandler, *The Art of Software Testing*, vol. 1. John Wiley & Sons, 2004.

[3] G. Rothermel, R. H. Untch, C. C. Chu, and M. J. Harrold, "Test case prioritization: an empirical study," in *Software Maintenance, 1999. (ICSM '99) Proceedings. IEEE International Conference on*, 1999, pp. 179–188.

[4] H. K. N. Leung, "Insights into Regression Testing," *Proceedings of the International Conference on Software Maintenance*, pp. 60–69, 1989.

[5] S. Elbaum, G. Rothermel, and J. Penix, "Techniques for improving regression testing in continuous integration development environments," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*, 2014, pp. 235–245.

[6] P. K. Chittimalli and M. J. Harrold, "Recomputing coverage information to assist regression testing," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 452–469, 2009.

[7] S. Yoo and M. Harman, "Regression Testing Minimisation, Selection and Prioritisation : A Survey," *Test Verif Reliab*, vol. 0, pp. 1–7, 2007.

[8] D. Jeffrey and N. Gupta, "Improving fault detection capability by selectively retaining test cases during test suite reduction," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 108–123, Feb. 2007.

[9] S. Elbaum, P. Kallakuri, A. G. Malishevsky, G. Rothermel, and S. Kanduri, "Understanding the effects of changes on the cost-effectiveness of regression testing techniques," *Journal of Software Testing, Verification and Reliability*, vol. 12, no. 2, pp. 65–83, 2003.

[10] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: a family of empirical studies," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 159–182, 2002.

[11] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. TR/SE-0401, p. 28, 2004.

[12] B. Kitchenham, O. P. Brereton, and D. B. et al., "Systematic literature reviews in software engineering �A systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009.

[13] Y. Singh, "Systematic Literature Review on Regression Test Prioritization Techniques Difference between Literature Review and Systematic Literature," *Informatica*, vol. 36, pp. 379–408, 2012.

[14] C. Catal and D. Mishra, "Test case prioritization: a systematic mapping study," *Software Quality Journal*, vol. 21, no. 3, pp. 445–478, 2012.

[15] A. Kumar and K. Singh, "A Literature Survey on test case prioritization," *Compusoft*, 2014.

[16] P. Kiran and K. Chandraprakash, "A Literature Survey on TCP-Test Case Prioritization using the RT-Regression Techniques," *Global Journal of*, 2015.

[17] P. Achimugu, A. Selamat, R. Ibrahim, and M. Naz, "A systematic literature review of software requirements prioritization research," *Information and Software*, vol. 56, pp. 568–585, 2014.

[18] S. Thomas, H. Hemmati, and A. Hassan, "Static test case prioritization using topic models," *Software Engineering*, 2014.

[19] S. Sampath, R. Bryce, and A. M. Memon, "A uniform representation of hybrid criteria for regression testing," *IEEE Transactions on Software Engineering*, vol. 39, no. 10, pp. 1326–1344, 2013.

[20] A. B. Sanchez, S. Segura, and A. Ruiz-Cortes, "A Comparison of Test Case Prioritization Criteria for Software Product Lines," *Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference*, pp. 41–50, 2014.

[21] L. Mei, W. K. Chan, T. H. Tse, S. Member, B. Jiang, and K. Zhai, "Preemptive Regression Testing of Workflow-Based Web Services," *IEEE Transactions on*, vol. 8, no. 5, pp. 740–754, 2015.

[22] C. Fang, Z. Chen, K. Wu, and Z. Zhao, "Similarity-based test case prioritization using ordered sequences of program entities," *Software Quality Journal*, vol. 22, no. 2, pp. 335–361, 2014.

[23] B. Miranda and A. Bertolino, "Scope-aided Test Prioritization, Selection and Minimization for

Software Reuse," *Journal of Systems and Software*, vol. 0, pp. 1–22, 2016.

[24] B. Korel, G. Koutsogiannakis, and L. H. Tahat, "Model-based test prioritization heuristic methods and their evaluation," *Proceedings of the 3rd international workshop on Advances in model-based testing - A-MOST '07*, pp. 34–43, 2007.

[25] R. Maheswari and D. Mala, "Combined Genetic and Simulated Annealing Approach for Test Case Prioritization," *Indian Journal of Science and Technology*, 2015.

[26] Y. Lou, D. Hao, and L. Zhang, "Mutation-based test-case prioritization in software evolution," *2015 IEEE 26th International Symposium on Software Reliability Engineering, ISSRE 2015*, pp. 46–57, 2016.

[27] F. Yuan, Y. Bian, Z. Li, and R. Zhao, "Epistatic Genetic Algorithm for Test Case Prioritization," *International Symposium on Search Based*, 2015.

[28] C. Catal, "On the application of genetic algorithms for test case prioritization: a systematic literature review," *Proceedings of the 2nd International Workshop on*, 2012.

[29] A. Kaur and S. Goyal, "A genetic algorithm for fault-based regression test case prioritization," *International Journal of Computer Applications*, vol. 32, no. 8, pp. 975–8887, 2011.

[30] W. Jun, Z. Yan, and J. Chen, "Test case prioritization technique based on genetic algorithm," *Internet Computing & Information*, 2011.

[31] S. Sabharwal, R. Sibal, and C. Sharma, "Prioritization of test case scenarios derived from activity diagram using genetic algorithm," *2010 International Conference on Computer and Communication Technology, ICCCT-2010*, pp. 481–485, 2010.

[32] K. Deb, S. Pratab, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NGSA-II," *IEEE Transactions on Evolutionary Computing*, vol. 6, no. 2, pp. 182–197, 2002.

[33] H. Do and G. Rothermel, "On the use of mutation faults in empirical assessments of test case prioritization techniques," *IEEE Transactions on Software Engineering*, vol. 32, no. 9, pp. 733–752, 2006.

[34] Z. Li, M. Harman, and R. M. Hierons, "Search Algorithms for Regression Test Case Prioritization," *IEEE Transactions on Software Engineering*, vol. 33, no. 4, pp. 225–237, 2007.

[35] S. Li, N. Bian, Z. Chen, and D. You, "A simulation study on some search algorithms for regression test case prioritization," *2010 10th International*, 2010.

[36] K. Solanki, Y. Singh, S. Dalal, and P. Srivastava, "Test Case Prioritization: An Approach Based on Modified Ant Colony Optimization," *Emerging Research in*, 2016.

[37] D. Gao, X. Guo, and L. Zhao, "Test case prioritization for regression testing based on ant colony optimization," *Software Engineering and Service*, 2015.

[38] T. Noguchi, H. Washizaki, and Y. Fukazawa, "History-Based Test Case Prioritization for Black Box Testing Using Ant Colony Optimization," *2015 IEEE 8th*, 2015.

[39] Y. Ledru, A. Petrenko, S. Boroday, and N. Mandran, "Prioritizing Test cases with string distances," *Automated Software Engineering*, vol. 19, no. 1, pp. 65–95, 2012.

[40] B. Jiang and W. K. Chan, "Input-based Adaptive Randomized Test Case Prioritization," *J Syst Softw*, vol. 105, no. C, pp. 91–106, 2015.

[41] S. Eghbali and L. Tahvildari, "Test Case Prioritization Using Lexicographical Ordering," *IEEE Transactions on Software Engineering*, vol. 5589, no. January, pp. 1–1, 2016.

[42] D. Di Nardo, N. Alshahwan, L. Briand, and Y. Labiche, "Coverage-based test case prioritization: An industrial case study," *Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013*, pp. 302–311, 2013.

[43] D. Nardo, N. Alshahwan, and L. Briand, "Coverage-based regression test case selection, minimization, and prioritization: a case study on an industrial system," *Software Testing*, 2015.

[44] D. Hao, L. Zhang, L. Zhang, G. Rothermel, and H. Mei, "A Unified Test Case Prioritization Approach," *ACM Trans Softw Eng Methodol*, vol. 24, no. 2, p. 10:1--10:31, 2014.

[45] D. Hao, L. Zhang, L. Zang, Y. Wang, X. Wu, and T. Xie, "To Be Optimal or Not in Test-Case Prioritization," *IEEE Transactions on Software Engineering*, vol. 42, no. 5, pp. 490–504, 2016.

[46] L. Zhang, D. Hao, L. Zhang, G. Rothermel, and H. Mei, "Bridging the gap between the total and additional test-case prioritization strategies," *Proceedings - International Conference on Software Engineering*, pp. 192–201, 2013.

[47] S.-Z. Haidry and T. Miller, "Using Dependency Structures for Prioritisation of Functional Test Suites," *IEEE Transactions on Software Engineering*, vol. 39, no. 2, pp. 1–1, 2012.

[48] C. R. Fang, Z. Y. Chen, and B. W. Xu, “Comparing logic coverage criteria on test case prioritization,” *Science China Information Sciences*, vol. 55, no. 12, pp. 2826–2840, 2012.

[49] R. C. Bryce, S. Sampath, J. B. Pedersen, and S. Manchester, “Test suite prioritization by cost-based combinatorial interaction coverage,” *International Journal of Systems Assurance Engineering and Management*, vol. 2, no. 2, pp. 126–134, 2011.

[50] J. Jones and M. Harrold, “Test-Suite Reduction and Prioritization for Modified Condition / Decision Coverage Georgia Institute of Technology,” *Test*, vol. 3, no. 3, pp. 101–195, 2003.

[51] D. Leon and A. Podgurski, “A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases,” *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, vol. 2003–Janua, pp. 442–453, 2003.

[52] R. Krishnamoorthi and S. A. Sahaaya Arul Mary, “Factor oriented requirement coverage based system test case prioritization of new and regression test cases,” *Information and Software Technology*, vol. 51, no. 4, pp. 799–808, 2009.

[53] S. Tahvili, W. Afzal, M. Saadatmand, and M. Bohlin, “Towards earlier fault detection by value-driven prioritization of test cases using fuzzy TOPSIS,” *Information Technology:*, 2016.

[54] E. L. G. Alves, P. D. L. Machado, T. Massoni, and M. Kim, “Prioritizing Test cases for early detection of refactoring faults,” *Software Testing Verification and Reliability*, vol. 26, no. 5, pp. 402–426, 2016.

[55] L. Mei *et al.*, “A Subsumption Hierarchy of Test Case Prioritization for Composite Services,” *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 658–673, 2015.

[56] Y. Wang, X. Zhao, and X. Ding, “An effective test case prioritization method based on fault severity,” *Software Engineering and Service*, 2015.

[57] Y. Qi, X. Mao, and Y. Lei, “Efficient automated program repair through fault-recorded testing prioritization,” *IEEE International Conference on Software Maintenance, ICSM*, pp. 180–189, 2013.

[58] B. Jiang, Z. Zhang, W. K. Chan, T. H. Tse, and T. Y. Chen, “How well does test case prioritization integrate with statistical fault localization?,” *Information and Software Technology*, vol. 54, no. 7, pp. 739–758, 2012.

[59] Y. T. Yu and M. F. Lau, “Fault-based test suite prioritization for specification-based testing,” *Information and Software Technology*, vol. 54, no. 2, pp. 179–202, 2012.

[60] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, “The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments,” *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 593–617, 2010.

[61] H. Srikanth, C. Hettiarachchi, and H. Do, “Requirements Based Test Prioritization Using Risk Factors,” *Inf Softw Technol*, vol. 69, no. C, pp. 71–83, 2016.

[62] T. Muthusamy, “A New Effective Test Case Prioritization for Regression Testing based on Prioritization Algorithm,” *International Journal of Applied Information Systems (IJAIS)*, vol. 6, no. 7, pp. 21–26, 2014.

[63] T. Ma, H. Zeng, and X. Wang, “Test case prioritization based on requirement correlations,” *2016 IEEE/ACIS 17th International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2016*, pp. 419–424, 2016.

[64] J. Badwal and H. Raperia, “Test Case Prioritization using Clustering,” *2013 IEEE Sixth International Conference*, pp. 488–492, 2013.

[65] C. Hettiarachchi, H. Do, and B. Choi, “Effective regression testing using requirements and risks,” *Proceedings - 8th International Conference on Software Security and Reliability, SERE 2014*, pp. 157–166, 2014.

[66] M. Yoon, “A Test Case Prioritization through Correlation of Requirement and Risk,” *Journal of Software Engineering and Applications*, vol. 5, no. 10, pp. 823–836, 2012.

[67] H. Srikanth, L. Williams, and J. Osborne, “System test case prioritization of new and regression test cases,” *Int’l Symp on Empirical Software Engineering*, vol. 0, no. c, pp. 62–71, 2005.

[68] H. Srikanth, M. Cashman, and M. B. Cohen, “Test case prioritization of build acceptance tests for an enterprise cloud application: An industrial case study,” *The Journal of Systems and Software*, vol. 119, pp. 122–135, 2016.

[69] C. T. Lin, C. D. Chen, C. S. Tsai, and G. M. Kapfhammer, “History-based test case prioritization with software version awareness,” *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, pp. 171–172, 2013.

[70] D. Marijan, A. Gotlieb, and S. Sen, “Test case prioritization for continuous regression testing:

An industrial case study," *Software Maintenance (ICSM),* 2013.
[71] A. Khalilian, M. Azgomi, and Y. Fazlalizadeh, "An improved method for test case prioritization by incorporating historical test case data," *Science of Computer*, 2012.
[72] J.-M. K. J.-M. Kim and a. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," *Proceedings of the 24th International Conference on Software Engineering ICSE 2002*, pp. 119–129, 2002.
[73] Y. C. Huang, K. L. Peng, and C. Y. Huang, "A history-based cost-cognizant test case prioritization technique in regression testing," *Journal of Systems and Software*, vol. 85, no. 3, pp. 626–637, 2012.
[74] C. Hettiarachchi, H. Do, and B. Choi, "Risk-based test case prioritization using a fuzzy expert system," *Information and Software Technology*, 2016.
[75] H. YOON and B. CHOI, "a Test Case Prioritization Based on Degree of Risk Exposure and Its Empirical Study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 21, no. 2, pp. 191–209, 2011.
[76] H. Stallbaum, A. Metzger, and K. Pohl, "An automated technique for risk-based test case generation and prioritization," *... on Automation of software test*, p. 67, 2008.
[77] M. Felderer and I. Schieferdecker, "A taxonomy of risk-based testing," *International Journal on Software Tools for Technology Transfer*, vol. 16, no. 5, pp. 559–568, 2014.
[78] E. Ufuktepe and T. Tuglular, "Automation Architecture for Bayesian Network Based Test Case Prioritization and Execution," *Computer Software and Applications*, 2016.
[79] X. Zhao, Z. Wang, X. Fan, and Z. Wang, "A Clustering-Bayesian Network Based Approach for Test Case Prioritization," *Computer Software and*, 2015.
[80] S. Mirarab and L. Tahvildari, "A Prioritization Approach for Software Test Cases Based on Bayesian Networks," *Fundamental Approaches to Software Engineering Springer Berlin Heidelberg*, vol. 4422, pp. 276–290, 2007.
[81] S. Mirarab and L. Tahvildari, "An Empirical Study on Bayesian Network-based Approach for Test Case Prioritization," *2008 International Conference on Software Testing, Verification, and Validation*, pp. 278–287, 2008.
[82] S. Elbaum, G. Rothermel, S. Kanduri, and A. G. Malishevsky, "Selecting a cost-effective test case prioritization technique," *Software Quality Journal*, vol. 12, no. 3, pp. 185–210, 2004.
[83] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: A survey," *Software Testing Verification and Reliability*, vol. 22, no. 2. pp. 67–120, 2012.
[84] A. B. Sanchez, S. Segura, and A. Ruiz-Cortes, "A Comparison of Test Case Prioritization Criteria for Software Product Lines," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, 2014, pp. 41–50.
[85] H. Srikanth, C. Hettiarachchi, and H. Do, "Requirements based test prioritization using risk factors: An industrial study," *Information and Software Technology*, vol. 69, pp. 71–83, 2016.
[86] M. J. Arafeen and H. Do, "Test Case Prioritization Using Requirements-Based Clustering," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, 2013, pp. 312–321.
[87] A. Shahbazi and J. Miller, "Black-Box String Test Case Generation through a Multi-Objective Optimization," *IEEE Transactions on Software Engineering*, vol. 42, no. 4, pp. 361–378, 2016.
[88] B. Jiang and W. Chan, "Input-based adaptive randomized test case prioritization: A local beam search approach," *Journal of Systems and Software*, 2015.
[89] S. Elbaum, A. Malishevsky, and G. Rothermel, *Prioritizing test cases for regression testing*. 2000.
[90] R. Fisher, "Statistical methods for research workers," 1925.
[91] G. Kapfhammer and M. Soffa, "Using coverage effectiveness to evaluate test suite prioritizations," *Proceedings of the 1st ACM international*, 2007.
[92] "Software-artifact Infrastructure Repository: Home." [Online]. Available: http://sir.unl.edu/portal/index.php. [Accessed: 20-Mar-2017].
[93] D. K. Yadav and S. Dutta, "Test case prioritization technique based on early fault detection using fuzzy logic," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1033–1036, 2016.
[94] A. Schwartz and H. Do, "Cost-effective regression testing through Adaptive Test Prioritization strategies," *Journal of Systems and Software*, vol. 115, pp. 61–81, 2016.
[95] J. A. Parejo, A. B. S??nchez, S. Segura, A. Ruiz-Cort??s, R. E. Lopez-Herrejon, and A. Egyed, "Multi-objective test case prioritization in highly configurable systems: A case study,"

Journal of Systems and Software, vol. 122, pp. 287–310, 2016.
[96] A. Marchetto, M. Islam, and W. Asghar, "A multi-objective technique to prioritize test cases," *IEEE Transactions*, vol. 42, no. 10, pp. 918–940, 2016.
[97] X. Xia, L. Gong, T.-D. B. Le, D. Lo, L. Jiang, and H. Zhang, "Diversity maximization speedup for localizing faults in single-fault and multi-fault programs," *Automated Software Engineering*, vol. 23, no. 1, pp. 43–75, Mar. 2016.
[98] M. Laali, H. Liu, M. Hamilton, M. Spichkova, and H. W. Schmidt, "Test Case Prioritization Using Online Fault Detection Information," Springer, Cham, 2016, pp. 78–93.
[99] W. Fu, H. Yu, G. Fan, and X. Ji, "Test Case Prioritization Approach to Improving the Effectiveness of Fault Localization," in *2016 International Conference on Software Analysis, Testing and Evolution (SATE)*, 2016, pp. 60–65.
[100] X.-Y. Zhang, D. Towey, T. Y. Chen, Z. Zheng, and K.-Y. Cai, "A random and coverage-based approach for fault localization prioritization," in *2016 Chinese Control and Decision Conference (CCDC)*, 2016, pp. 3354–3361.

Appendix

This appendix section contains Table X1 – Table X5

Table X1: Result Quality Scores of Selected Studies.

Paper Refs.	Q1	Q2	Q3	Q4	Q5	Score
Rothermel et al., 1999 [3]	1	1	0.5	0.5	1	4
Yoo, S., & Harman, M. 2012 [7]	1	0	1	0	1	3
Singh et al., 2012 [13]	1	0	1	0	1	3
Thomas et al., 2014 [18]	1	1	1	0.5	1	4.5
Sampath et al., 2013 [19]	1	1	1	0.5	1	4.5
Sanchez et al., 2014 [20]	1	1	0.5	0.5	1	4
Mei et al., 2015 [21]	1	1	1	0.5	1	4.5
Fang et al., 2014 [22]	1	1	1	0.5	1	4.5
Miranda & Bertolino, 2016 [23]	1	1	1	0.5	1	4.5
Korel et al., 2007 [24]	1	1	0.5	0.5	1	4
Maheswari et al.,2015 [25]	1	1	1	0.5	1	4.5
Lou et al., 2015 [26]	1	1	0.5	0.5	1	4
Yuan et al., 2015 [27]	1	1	0.5	0.5	1	4
Catal, C. 2012 [28]	1	1	0.5	0.5	1	4
Kaur, A., & Goyal, S. 2011 [29]	1	1	1	0.5	1	4.5
Jun et al., 2011 [30]	1	1	0.5	0.5	1	4
Sabharwal et al., 2010 [31]	1	1	0.5	0.5	1	4
Do et al., 2006 [33]	1	1	1	0.5	1	4.5
Deb et al., 2002 [32]	1	1	1	0.5	1	4.5
Li et al., 2007 [34]	1	1	1	0.5	1	4.5
Li et al., 2010 [35]	1	1	0.5	0.5	1	4
Solanki et al., 2016 [36]	1	1	0.5	0.5	1	4
Gao et al., 2015 [37]	1	1	0.5	0.5	1	4
Noguchi et al., 2015 [38]	1	1	0.5	0.5	1	4
Ledru et al., 2012 [39]	1	1	1	0.5	1	4.5
Jiang et al., 2015 [40]	1	1	1	0.5	1	4.5