International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

Regression Test Case Prioritization Frameworks: Challenges and Future Directions

Bakr Ba-Quttayyan, Haslina Mohd, Yuhanis Yusof

Abstract: ***Regression testing is a necessary maintenance activity in the software industry where modified software programs are revalidated to make sure that changes do not adversely affect their behavior. Test case prioritization (TCP) is one of the most effective methods in regression testing whereby test cases are rescheduled in an appropriate order for execution to increase test effectiveness in meeting some performance goals such as increasing the rate of fault detection. This paper explores efforts that have been carried out in relation to TCP frameworks. Through the review of related literature, ten existing frameworks were identified, classified and reviewed whereby two are Bayesian network-based, five are multi-objective, while the rest are varied in terms of aspects and purposes. Accordingly, this study analyzes those frameworks based on their proposed year, TCP factors, number of test cases used, evaluations metric and criteria as well as experimental subjects. The results showed that the stated frameworks are not integrated with nature-inspired algorithms as enhancing optimization techniques while several others were insufficiently evaluated according to stated evaluation criteria and metrics for the effective and practical testing process. There is also a scarcity of frameworks that focus on regression test efficiency. This study indicates the need for further research into the topic to enhance TCP frameworks that focus on several directions for practical considerations in this field such as evaluation issues, specific knowledge dependency, and objective deviation. At the end of this study, several future directions such as nature-inspired algorithms assistance are proposed, and a number of limitations are identified and highlighted.***

Keywords**: *Regression testing, test case prioritization, test effectiveness, test efficiency, test frameworks.***

I. INTRODUCTION

Regression testing is a necessary maintenance activity in the software industry where modified programs are revalidated to make sure that changes do not adversely affect their behavior. Naturally, regression testing is carried out by reusing available test cases of earlier versions of the current software system in addition to newly created test cases for testing new features [1], [2]. In practice, frequent changes can lead to the enlargement of three aspects: size, test suites and complexity of the developed software which render the regression testing to be costly and time-consuming [2]. Due to cost and time constraints, regression testing can be performed in three different ways i.e. using test case selection (TCS), test case prioritization (TCP) as well as a hybrid approach [3]. TCS entails selecting a subset of the existing tests to revalidate the modified software system. The main purpose of the selection process is to mimic the testing cost without affecting its effectiveness [4]. Test selection has three categories: *1) safe* i.e. the selection of test cases that provide different outputs for a modified program as compared to the base version, *2) coverage* i.e. the selection of test cases based on coverage criteria, and *3) reduction* or *minimization* whereby redundant test cases are discarded [5], [6]. Meanwhile, the TCP method reorders test cases in a manner that increases test effectiveness without omitting any test case. A hybrid method combines TCP with other selection methods. In comparison to the selection and hybrid methods, test prioritization is considered as the most effective method in regression testing due to the higher possibility of finding hidden errors, as no test case is removed [7], [8].

Since its introduction in 1997, TCP has been one of the most effective methods for performing regression testing [9]. In TCP, test cases are rescheduled for execution in an appropriate order so as to increase test effectiveness in meeting some performance goals such as increasing the rate of faults detection [2], [10], [11]. In terms of performance, TCP outperforms other regression testing approaches, such as the test selection and hybrid approach, as TCP accelerates fault detection, mimics bug fixing time and avoids test omission drawbacks [10]–[13]. Accordingly, several efforts have been proposed in literature in the domain of TCP. Several studies demonstrated frameworks that have been introduced to increase the effectiveness or reduce the cost of performing TCP. In this article, such frameworks on TCP are presented and discussed. With respect to the aforementioned, this paper attempts to answer the following research questions:

- How many frameworks have been proposed for TCP?
- What TCP factors are used in these frameworks?
- How have these frameworks been evaluated?
- What measurement was used to evaluate these frameworks in terms of test efficiency and effectiveness?

Manuscript published on November 30, 2019.
* Correspondence Author
Bakr Ba-Quttayyan*, Human-Centered Computing (HCC) research lab, School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010, Sintok, Kedah, Malaysia. Email: baksam1@gmail.com
Haslina Mohd, Human-Centered Computing (HCC) research lab, School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010, Sintok, Kedah, Malaysia. Email: haslina@uum.edu.my
Yuhanis Yusof, School of Computing, College of Arts and Sciences, Universiti Utara Malaysia, 06010, Sintok, Kedah, Malaysia. Email: yuhanis@uum.edu.my

The rest of this paper entails a section describing the current TCP frameworks, a discussion section and finally the conclusion.

II. METHODOLOGY

Peffers, Tuunanen, Rothenberger and Chatterjee [14] define methodology as the overall approach which consists of the conceptual principles, practice rules and process for conducting a research. Accordingly, this paper carries out an exploration study for frameworks proposed in the TCP domain which entails several steps. The first step entail conducting a search in a number of databases such as: IEEE Xplore, Springer link and Google scholar for collecting studies on TCP frameworks. The second step is collecting and classifying these studies. Third, analyzing, discussing and reporting the results of this process. Ultimately, ten frameworks were identified, analyzed and reported as shown in the following sections.

III. CURRENT TCP FRAMEWORKS

Several frameworks have been proposed in the field of regression test case prioritization since 2007. Through the review of related literature, ten frameworks for TCP were identified in this study. Two of the frameworks were developed based on the Bayesian Networks (BN), five were multi-objective, while the remainder of the frameworks were aspect-based.

A. BN-based Frameworks

Mirarab and Tahvildari [15], [16] introduced the first TCP framework based on the Bayesian Networks which rely on the probability theory. To prioritize test cases, this framework extracts information from three artifacts namely source code changes, software fault-proneness, and test coverage data, and incorporates those information to build a BN model which uses probabilistic inference for prioritizing test cases based on their probability of success. An optimized set of test cases which was ranked based on their fault prediction capability was produced as an output of this framework. The weighted average percentage of faults detected (APFD) metric was used to evaluate this framework, whilst the data for experiment was derived from the Software-artifact Infrastructure Repository (SIR). The framework was subsequently compared to three other techniques. The Java programming language was used to develop the framework which was then evaluated using 5 Java programs provided in the SIR repository namely: *ant*, *xml-security*, *jmeter*, *nanoxml*, and *galileo*. The techniques used for comparison were: BN-based (with variations: BN and BNA with feedback incorporated), original ordering as control technique, and two coverage-based conventional techniques (MC and MCA with feedback). According to Mirarab [17], this framework achieved an average of 90% of the APFD metric, with an average of 3.34 seconds running time for the BN algorithm and an average of 127.44 seconds running time for the BNA algorithm. Mirarab and Tahvildari [16] indicated that within the same framework, the BNA technique is 3 to 100 times slower than the BN technique. However, it is important to note that the framework was evaluated based on hand-seeded faults instead of real faults, apart from using feedback (i.e. BNA technique) which always increases the running time as indicated by Mirarab [17].

An extension of the previous works was proposed by [18], which incorporates the similarities between the test cases. To tackle such similarities, a code coverage-based clustering approach was used. The researchers indicated that the CBN approach outperformed both the BN and BNA approaches, with recorded APFD achievements of 86.9, 81.2 and 83.7 respectively. Ufuktepe and Tuglular [19] also suggested an automated platform and tool for implementing the first BN-based framework which was introduced by Mirarab and Tahvildari [15].

B. Multi-objective Frameworks

As mentioned earlier, a multi-objective TCP has more than one objective in the prioritization process. The multi-objective TCP framework proposed by Siddik and Sakib [20] combines different software artifacts such as requirements coverage, design diagrams and code coverage, clustering requirements and test cases and subsequently mapping customers' requirements with test cases to produce the final order of test cases [21], [22]. However, this framework ignores historical data regarding faults as well as the threat of detecting the same faults by different clusters of test cases [23].

In terms of time-constraint, Marijan [24] introduced a multi-perspective (MP) TCP framework which integrates three different perspectives: business, performance and technical. Three objectives and a constraint were used namely failure impact, test execution time, failure frequency and cross-functionality (for test coverage). This framework was evaluated using three android-based software systems and compared to manual practice. The evaluation metric used is cost-cognizant weighted average percentage of faults detected ($APFD_C$) which was proposed by Elbaum, Malishevsky and Rothermel [25], and of which recorded a 19.59% achievement. This work is indicated to be promising by Ricken and Dyck [26] due to its industrial aspect.

Finally, the dissimilarity clustering framework was introduced by [23], which attempts to prioritize test cases based on their dissimilarity, code coverage and historical failure data. This framework was evaluated using the APFD metric, involving three projects from the *Defects4j* datasets for experiments, and compared with three different prioritization scenarios namely: untreated (original), random, and similarity ordering. This framework recorded an average APFD of 88.54%, which outperforms the compared orderings.

Bian, Li, Gao and Zhao [7] proposed a concrete hyper-heuristic framework for prioritizing test cases. The main purpose of this framework is to select, among several algorithms in the repository, the appropriate algorithm for different TCP scenarios. For evaluation, the average percentage of statement coverage (APSC) metric was used to evaluate the effectiveness of the framework, and the effective execution time (EET) formula to compute test efficiency.

This study reported an average APSC of 96.34% for statement coverage and 738.87 as an average of the EET when 100% statement coverage was achieved for several iterations compared to 18734.4 seconds of the average execution time with reduction of around 96% of the actual time.

This study argues that the execution time for this framework is always shorter than the multi-objective PSO algorithm [7].

Finally, the Graphite framework was proposed by Azizi and Do [27] for prioritizing test cases using greedy graph model. The prioritization process attempt is based on code coverage, test case dissimilarity and test cost, i.e. test execution time. The framework was evaluated using APFD in a time-constraint environment (25%, 50%, and 75%), which achieved an average of 85% of fault detection rate during several executions and compared with three greedy-based techniques. This framework intends to improve the efficiency and effectiveness of test prioritization. Moreover, this framework uses execution time (ET) to scale test efficiency as well as APFD to measure the test effectiveness. In terms of test efficiency, the Graphite framework obtained an average of 1.400 seconds of execution time in detecting all faults as opposed to 2.000 and 3.500 seconds by the compared techniques. This means that it reduces around 30% to 60% of test execution time which is an indicator of test efficiency improvement. However, based on their experiment, this framework needs more empirical evaluation with different applications and techniques.

C. Other Frameworks

This section specifies frameworks that ignore test cases as either a basis for prioritization process or for the purpose of regression testing.

Praba and Mala [28] proposed a component-based framework for real-time systems which prioritizes critical components according to their dependency metrics. The framework was developed and tested using Java and evaluated based on internal and external metrics. This framework was then compared to two methods: full regression and unit regression. In terms of time consumption i.e. in seconds, this framework was found to be better than the full regression method. However, there is a possibility of the occurrence of hazards because some components were skipped during testing.

Sampath, Bryce, Jain, and Manchester [29] developed a framework for prioritizing user-sessions test cases according to HTTP request for web-based systems. This framework was implemented using Java and C languages. However, the evaluation for this framework did not specify the metrics or results of comparison with others.

Lastly, Qian and Zhou [30] introduced a framework for prioritizing test cases according to their memory leak (ML) ability using a prediction model built specifically for android applications. This framework used code-level features for prediction, Android virtual device for experiment, and nine subjects for evaluation. The prioritized test cases, according to this method, were categorized into three groups: best, median, and worst rank. However, the execution time and evaluation metric were not specified for this work.

These three frameworks do not reflect the purpose of regression testing in terms of faults detection and prioritized test cases. Table I below summarizes the above-mentioned TCP frameworks in chronological order, together with their respective information including authors, objectives used in

Table- I: The Current TCP Frameworks

No.	Title	Source	Year	TCP factors	Eval. metric	Achievement	Comparison	# of TCs	Experimental subjects
F1	BN	[15], [16]	2007	Fault prediction	APFD	90%	4 techniques	2166	Software-artifact Infrastructure Repository (SIR)
F2	CCA	[28]	2011	Critical components	Dependency metrics	99%	2 techniques	Nil	Vehicle Management System
F3	CPUT	[29]	2011	User-session	Nil	Nil	Nil	173	Schoolmate web application
F4	RDCC	[20]	2014	1) Req 2) Design 3) Code	Nil	Nil	Nil	10	Academic Time Synchronization Project (ATSP)
F5	MP	[24]	2015	1) Business 2) Performance 3) Technical	$APFD_C$	19.59%	With Manually	1854	3 Mobile Apps
F6	CBN	[18]	2015	Fault prediction	APFD	86.9%	4 techniques	Nil	Software-artifact Infrastructure Repository (SIR)
F7	ML	[30]	2016	Memory leaks detection	Nil	2.6 - 27.3% (Best)	Nil	242 (20)	9 Mobile Apps
F8	CDTC	[23]	2017	1) Total # of faults detected 2) Clustering dissimilar TC	APFD	88.54%	3 techniques	12377	Defects4j

F9	HH[j]	[7]	2018	1) Structure coverage (statement). 2) Effective execution time.	APSC, EET	96.34%	4 techniques	28422	8 subjects (5 Java from Github, 3 C++ from SIR and Google).
F10	Graphite	[27]	2018	1) Code coverage. 2) Test cases dissimilarity. 3) Test cost (test execution time).	APFD, ET	85%	3 techniques	4130	4 subjects (2 from SIR).

prioritization, evaluation metrics, achievements gained, number of test cases used during the experiments as well as experimental subjects.

IV. RESULTS AND DISCUSSION

The preceding section had provided a rich description of the existing TCP frameworks. The following subsections present the evaluation and limitations of those TCP frameworks.

A. Evaluation of TCP Frameworks

Based on the discussion in the previous sections, there are several criteria for the evaluation of TCP frameworks including the evaluation metrics, achievement according to the said metric, and comparison with other techniques in the field. Table I presents the four frameworks that were evaluated for their effectiveness using a weighted average percentage of faults detected (APFD) metric namely F1, F6, F8, and F10. The APFD is a well-known metric developed by Rothermel *et al.* [8] which has since became the most popular metric for evaluating the effectiveness of TCP whereby more than 40% of current studies in the field of TCP employ APFD [31]. Moreover, the calculated values for APFD range from 0 up to 100, as the higher value indicates the better (faster) fault detection rates and can be calculated as [1], [32]:

$$APFD = 1 - \frac{TF_1 + TF_2 + ... + TF_m}{n \times m} + \frac{1}{2n} \quad (1)$$

Where T is a test suite containing n test cases, F a set of m faults revealed by T, m total number of faults detected by given test suite, n the total number of test cases in T, and TF_i the position of the first test case detects fault i.

Furthermore, F5 was evaluated using the $APFD_C$ metric, which is derived from the APFD, but incorporates the aspects of cost and fault severity and is calculated as follows [1], [25]:

$$APFD_C = \frac{\sum_{i=1}^{m}\left(f_i \times \left(\sum_{j=TF_i}^{n} t_j - \frac{1}{2}t_{TF_i}\right)\right)}{\sum_{i=1}^{n} t_i \times \sum_{i=1}^{m} f_i} \quad (2)$$

Where t_j is the cost of test case, and f_i fault severity.

Table- II: Evaluation Criteria for the TCP Frameworks

No.	Source	Evaluation Criteria			
		Metric	*Achievement*	*Comparison*	*Multi-objective*
F1	[15], [16]	✓	✓	✓	×
F2	[28]	✓	✓	✓	×
F3	[29]	×	×	×	×
F4	[20]	×	×	×	✓
F5	[24]	✓	✓	×	✓
F6	[18]	✓	✓	✓	×
F7	[30]	×	✓	×	×
F8	[23]	✓	✓	✓	✓
F9	[7]	✓	✓	✓	✓
F10	[27]	✓	✓	✓	✓

Next, F9 was evaluated using a weighted average percentage of statement coverage (APSC), which is also derived from the APFD for scaling statement coverage and is calculated using the following equation [7], [33]:

$$APSC = 1 - \frac{TS_1 + TS_2 + ... + TS_m}{n \times m} + \frac{1}{2n} \quad (3)$$

Where TS_m represents test case that covers statement m.

Lastly, F2 was evaluated using the dependency metrics whilst F3, F4 and F7 encountered several evaluation issues whereby no metric/measurement, achievement or comparisons were specified. Table II below specifies the evaluation criteria for those frameworks.

B. Limitations of the TCP Frameworks

As presented in the table above, there is a gap in the evaluation of the metric, achievement, comparison and used test cases apart from the fact that most of the current frameworks have single objectives. The level of effectiveness recorded by these studies reaches almost up to 90% of the APFD which indicates that further enhancements are needed, and that more artifacts can be incorporated in the prioritization process. Other deficiencies are discussed in the following points:

1) ***Reliance on code.*** Frameworks that depend on code such as F1, F4, F6, F7, and F8, F9 and F10, cannot work in the absence of a software code which mimics the generality of the

framework. With the development of COTS, such frameworks will become impractical in the absence of the code.

2) ***Reliance on specific knowledge.*** F1 and F6 are BN-based; this type of technique requires specific knowledge of probability and is difficult to build and use. Additionally, BN-based techniques are time consuming during execution [10].
3) ***Objective deviation.*** The main objective of test case prioritizations is fault detection. Other objectives may lead to a decrease in fault detection ability especially when there is conflict with the main objective. Due to this matter, proper balance is needed specifically in multi-objective TCPs [9]. In this regard, not all these proposed frameworks are concentrated, either directly or indirectly, towards fault detection as the main objective for TCP; some of them have not been evaluated based on this objective. Hence, frameworks such as F2, F3 and F7 have no direct or indirect concerns towards fault detection ability.
4) ***Evaluation issues.*** As shown in Table 1 and 2, F3, F4 and F7 have evaluation issues i.e. lack of sufficient evaluation.
5) ***Nature-inspired assistance.*** None of the reported TCP frameworks are built based on nature-inspired algorithms which are more promising and well-known for solving multi-objective optimization problems [10].
6) ***Test efficiency.*** As stated before, only two multi-objective frameworks, i.e., F9 and F10 were concerned about test efficiency improvements. This indicates the need for more research on efficiency enhancements for regression test case prioritization.
7) ***Efficiency measurement.*** Frameworks concerning test efficiency such as F9 and F10 used the calculation of the actual time to scale the efficiency. For more enhancements, other measurements are used such as the ISO/IEC 25023:2016(E) [34] standard for measurement of system and software product quality which provides formulas that can be used for measuring test efficiency.

Due to the above demerits, there is a need to enhance TCP frameworks in terms of their evaluation, artifacts, efficiency and objectives, by incorporating nature-inspired algorithms and more for improving this domain.

V. CONCLUSION

Test case prioritization is one of the most effective methods for performing regression testing whereby test cases are rescheduled in an appropriate order to increase test effectiveness in meeting some performance goals such as increasing the rate of fault detection. In this study, the frameworks for test case prioritization were identified, classified and reviewed. Ten identified frameworks were critically surveyed in terms of purpose, objectives, experiments, and evaluation. Moreover, two TCP frameworks are Bayesian network-based, five are multi-objective, while the rest are varied in terms of aspects and purposes. Based on the deviations and observations, this study indicates the need for further research to enhance test case prioritization via the development of frameworks that focus on several directions for practical considerations in this field. Besides that, there is a need for more empirical studies in the evaluation of current TCP techniques. Furthermore, most of these frameworks and techniques focus primarily on test effectiveness while ignoring the test efficiency of test case prioritization. Future studies may bridge some of these demerits in the effort to enhance TCP.

REFERENCES

1. H. Do, "Recent Advances in Regression Testing Techniques," in Advances in Computers, vol. 103, Elsevier, 2016, pp. 53–77.
2. H. Hemmati, "Advances in Techniques for Test Prioritization," in Advances in Computers, 1st ed., Elsevier Inc., 2018, pp. 1–37.
3. B. Ba-Quttayyan, H. Mohd, and F. Baharom, "Regression Testing Systematic Literature Review – A Preliminary Analysis," Int. J. Eng. Technol., vol. 7, No 4.19, no. Special Issue 19, pp. 418–424, 2018.
4. R. Kazmi, D. N. A. Jawawi, R. Mohamad, and I. Ghani, "Effective Regression Test Case Selection: A Systematic Literature Review," ACM Comput. Surv., vol. 50, no. 2, pp. 1–32, May 2017.
5. S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," Softw. Testing, Verif. Reliab., vol. 22, no. 2, pp. 67–120, Mar. 2012.
6. G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," IEEE Trans. Softw. Eng., vol. 22, no. 8, pp. 529–551, 1996.
7. Y. Bian, Z. Li, J. Guo, and R. Zhao, "Concrete hyperheuristic framework for test case prioritization," J. Softw. Evol. Process, vol. 30, no. 11, p. e1992, Nov. 2018.
8. G. Rothermel, R. H. Untcn, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," IEEE Trans. Softw. Eng., vol. 27, no. 10, pp. 929–948, 2001.
9. D. Hao, L. Zhang, and H. Mei, "Test-case prioritization: achievements and challenges," Front. Comput. Sci., vol. 10, no. 5, pp. 769–777, Jul. 2016.
10. M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, "Test case prioritization approaches in regression testing: A systematic literature review," Inf. Softw. Technol., vol. 93, pp. 74–93, Jan. 2018.
11. P. Saraswat, A. Singhal, and A. Bansal, "A Review of Test Case Prioritization and Optimization Techniques," in Software Engineering, vol. 731, 2019, pp. 507–516.
12. P. Mahali and D. P. Mohapatra, "Model based test case prioritization using UML behavioural diagrams and association rule mining," Int. J. Syst. Assur. Eng. Manag., vol. 9, no. 5, pp. 1063–1079, Oct. 2018.
13. N. Gupta, A. Sharma, and M. K. Pachariya, "An Insight Into Test Case Optimization: Ideas and Trends With Future Perspectives," IEEE Access, vol. 7, pp. 22310–22327, 2019.
14. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," J. Manag. Inf. Syst., vol. 24, no. 3, pp. 45–77, Dec. 2007.
15. S. Mirarab and L. Tahvildari, "A Prioritization Approach for Software Test Cases Based on Bayesian Networks," in Fundamental Approaches to Software Engineering, vol. 4422, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 276–290.
16. S. Mirarab and L. Tahvildari, "An Empirical Study on Bayesian Network-based Approach for Test Case Prioritization," in 2008 International Conference on Software Testing, Verification, and Validation, 2008, pp. 278–287.
17. S. Mirarab, "A Bayesian Framework for Software Regression Testing," 2008.
18. X. Zhao, Z. Wang, X. Fan, and Z. Wang, "A Clustering-Bayesian Network Based Approach for Test Case Prioritization," in 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015, vol. 3, pp. 542–547.
19. E. Ufuktepe and T. Tuglular, "Automation Architecture for Bayesian Network Based Test Case Prioritization and Execution," in 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016, vol. 2, pp. 52–57.
20. M. S. Siddik and K. Sakib, "RDCC: An effective test case prioritization framework using software requirements, design and source code collaboration," in 2014 17th International Conference on Computer and Information Technology (ICCIT), 2014, pp. 75–80.

21. M. Abdur, M. Abu, and M. Saeed, “Prioritizing Dissimilar Test Cases in Regression Testing using Historical Failure Data,” Int. J. Comput. Appl., vol. 180, no. 14, pp. 1–8, Jan. 2018.
22. S. Wang, J. Nam, and L. Tan, “QTEP: quality-aware test case prioritization,” in Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017, 2017, pp. 523–534.
23. M. Abu Hasan, M. Abdur Rahman, and M. Saeed Siddik, “Test Case Prioritization Based on Dissimilarity Clustering Using Historical Data Analysis,” in International Conference on Information, Communication and Computing Technology ICICCT 2017, vol. 750, S. Kaushik, D. Gupta, L. Kharb, and D. Chahal, Eds. Singapore: Springer Singapore, 2017, pp. 269–281.
24. D. Marijan, “Multi-perspective Regression Test Prioritization for Time-Constrained Environments,” in 2015 IEEE International Conference on Software Quality, Reliability and Security, 2015, pp. 157–162.
25. S. Elbaum, A. Malishevsky, and G. Rothermel, “Incorporating varying test costs and fault severities into test case prioritization,” in Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001, 2001, pp. 329–338.
26. K. Ricken and A. Dyck, “A Survey on Multi-objective Regression Test Optimization,” in Full-scale Software Engineering/The Art of Software Testing, 2017, pp. 32–37.
27. M. Azizi and H. Do, “Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization,” in 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018, pp. 245–251.
28. M. R. Praba and D. J. Mala, “Critical component analyzer A novel test prioritization framework for component based real time systems,” in 2011 Malaysian Conference in Software Engineering, 2011, pp. 281–286.
29. S. Sampath, R. C. Bryce, S. Jain, and S. Manchester, “A tool for combination-based prioritization and reduction of user-session-based test suites,” in 2011 27th IEEE International Conference on Software Maintenance (ICSM), 2011, no. c, pp. 574–577.
30. J. Qian and D. Zhou, “Prioritizing Test Cases for Memory Leaks in Android Applications,” J. Comput. Sci. Technol., vol. 31, no. 5, pp. 869–882, Sep. 2016.
31. A. Bajaj and O. P. Sangwan, “A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms,” IEEE Access, vol. 7, pp. 126355–126375, 2019.
32. S. Elbaum, P. Kallakuri, A. Malishevsky, G. Rothermel, and S. Kanduri, “Understanding the effects of changes on the cost-effectiveness of regression testing techniques,” Softw. Test. Verif. Reliab., vol. 13, no. 2, pp. 65–83, 2003.
33. Z. Li, M. Harman, and R. M. Hierons, “Search Algorithms for Regression Test Case Prioritization,” IEEE Trans. Softw. Eng., vol. 33, no. 4, pp. 225–237, Apr. 2007.
34. I. O. for S. ISO, ISO/IEC 25023:2016(E) Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of system and software product quality, First Ed. Geneva, Switzerland.: International Organization for Standardization [ISO], 2016.

AUTHORS PROFILE

Bakr Ba-Quttayyan has a Master's degree in Software Engineering from Universiti Teknologi Malaysia (UTM) and a Bachelor's degree in Electronic Commerce from Al-Ahgaff University, Yemen. Currently, he is a PhD candidate at Universiti Utara Malaysia (UUM). His research interests include software engineering, programming, systems analysis and design, quality and software testing (e.g., test design, test planning, test documentation, agile testing, testing tools, test automation, and regression testing).

Haslina Mohd is an associate professor at Universiti Utara Malaysia (UUM). She has a doctorate degree in Health Informatics from Universiti Sains Malaysia (USM), and a Master's in Science from University of Liverpool. Her research interests include Software Quality and Software Testing. Currently, she is a research fellow at the Human Centered Computing (HCC) research lab, UUM.

Yuhanis Yusof is an associate professor at Universiti Utara Malaysia (UUM). She has a PhD in Computer Science from Cardiff University, United Kingdom. She also holds a MSc. degree in Computer Science from Universiti Sains Malaysia (USM), and Bachelor of Information Technology from UUM. Her research interest is broadly in data analytics and management for large scale computing. This includes data mining (discovering patterns of interest from data), information retrieval and optimization. Currently, she is involved in several projects relating to Machine Learning and Swarm Intelligence (e.g. Artificial Bee Colony, Firefly algorithm, Grey Wolf optimizer, Cuckoo Search and Bat algorithm).