Trends in prioritization of test cases: 2017-2019

M. del Carmen de Castro-Cabrera
Department of Computer Science,
University of Cádiz
Cádiz, Spain
maricarmen.decastro@uca.es

Antonio García-Dominguez
SEA research group,
EAS, Aston University
Birmingham, United Kingdom
a.garcia-dominguez@aston.ac.uk

Inmaculada Medina-Bulo
Department of Computer Science,
University of Cádiz
Cádiz, Spain
inmaculada.medina@uca.es

ABSTRACT

A core task in software testing is the design of test suites. Large test suites may take too long to run frequently, and test case prioritization (TCP) techniques have been proposed to speed up the detection of faults. These techniques have become increasingly popular and the number of publications has grown in recent years. Surveys have covered most of the techniques, but the latest included only publications until 2016: interest is growing, and new proposals have been developed in the last three years. This paper aims to complete that survey by providing the latest developments in TCP to respond to this growing interest. Specifically, we use the taxonomy proposed by Khatibsyarbin et al. on the most important publications from 2017 to the present day (2019). All in all, we found 320 papers in this period about test case prioritization. The results show that the main techniques used are search-, coverage- and similarity-based.

KEYWORDS

Test case prioritization, regression testing, systematic literature review, TCP, software testing.

ACM Reference Format:
M. del Carmen de Castro-Cabrera, Antonio García-Dominguez, and Inmaculada Medina-Bulo. 2020. Trends in prioritization of test cases: 2017-2019. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341105.3374036

1 INTRODUCTION AND MOTIVATION

Software testing is a time-consuming phase in the software development process. In fact, it can be the most expensive phase in all this process [30]. Therefore, much effort has been put into trying to reduce that time.

A significant aspect in software testing is the selection of a test suite, as it impacts the quality of the results and the time required for execution. This is even more important for large suites. Therefore, it is a question of choosing a set that guarantees reliable software testing. Once the test suite has been defined, test case prioritization techniques are applied to make the process more efficient.

In the literature, we can find numerous test case prioritization (TCP) approaches, as the survey by Khatibsyarbini [18] shows. In particular, this survey is cited by many other papers:

- In Scopus it is cited by 6 papers [3, 10–12, 15, 17] in 2018.
- In Google Scholar it is cited by 17 recent papers [3, 6, 7, 10–12, 12, 13, 15–17, 21, 28, 29, 31, 44, 46] (14 in 2018 and 3 in 2019).

However, the survey by Khatibsyarbini is from 2017, and there has been much work since then in TCP. This paper aims to complete this survey with the papers published since then on TCP, responding to the growing interest in the last years [19]. The systematic literature review (SLR) methodology proposed by Kitchenham will be used [20].

This paper is structured as follows. Section 3 presents the research method used in this work including five subsections: Subsection 3.1 selects the research questions to take into account in this work; Subsection 3.2 enumerates the repositories used; Subsection 3.3 chooses the publication search strategy; Subsection 3.4 establishes the inclusion and exclusion criteria; and, Subsection 3.5 presents the data synthesis and extraction. Then, Section 4 highlights the main results obtained, and Section 5 outlines their threats to validity. Finally, Section 6 presents the conclusions and future lines of research.

2 BACKGROUND

There are several approaches that can be followed when the time available to run regression test cases is limited: the three most common ones are test case minimisation, test case selection and test case prioritisation. Elbaum et al. provided clear definitions of test case selection (TCS, also known as regression test selection or RTS) and test case prioritization (TCP) [8]. Let P be a program, let P' be a modified version of P, and let T be a test suite for P. Regression testing is concerned with validating P'. Regression test selection (RTS) techniques select from test suite T a subset T' that contains test cases that are important to re-run. In contrast, test case prioritization (TCP) techniques reorder the test cases in T so testing objectives can be met sooner. Because TCP techniques do not themselves discard test cases, they can avoid the drawbacks that can occur when regression test selection cannot achieve safety. On the other hand, test case minimization techniques (TCM) seek to reduce the size of a test suite by eliminating redundant test cases from the test suite. Minimization is sometimes also called *test suite reduction*, meaning that the elimination is permanent [45]. Therefore, in this review, we will refer only to TCP techniques, discarding selection and minimization techniques.

Before we begin our review, let us recall the main lines of Khatibsyarbini's review [18]. The objective of this work was to examine and classify the current proposals on prioritization of test cases. Several repositories were searched for certain keywords and inclusion and exclusion criteria were defined for the results found. The papers were classified into journal articles, conference papers, symposiums and workshops.

Subsequently, a more in-depth study was carried out in which five criteria were applied to evaluate the quality of the work: precision of the objectives, clarity in the description of the approach, adequacy of the design of the strategy of the experiments, whether it has been applied to a case study or a controlled experiment has been carried out, and whether it represents an improvement in the academic environment.

The selected works were grouped into seven categories (dimensions in the original paper): six main ones, and one that grouped the rest of the approaches with fewer publications. These categories are briefly described below:

- *Coverage-based*: in these works, the code is inspected directly. Mainly, function, branch and sentence coverage is used.
- *Requirement-based*: these approaches take into account information on requirements, either through prioritization of requirements for testing, or considering algorithms based on traceability, completeness, the impact of a failure on requirements, changes in requirements, configurable priority and vision of developers.
- *Risk-based*: they consider information on the risks of the program. It also refers to the priority requirements on risk values.
- *Search-based*: they compute priorities through search-based algorithms such as genetic algorithms, greedy search, or ant colonies.
- *Fault-based*: they produce sequences of test cases to detect targeted faults. The aim is to prioritise the test cases most likely to detect errors and, at the same time, those whose execution takes less time.
- *History-based*: these approaches use historical data such as execution history or change information from past test executions to prioritise them.
- *Other-based*: this catch-all category includes a number of techniques. For instance, approaches based on Bayesian networks (which foresee the possibility that each test case will find an error), or cost-aware approaches (estimating the error ratio in unit tests with genetic algorithms).

With respect to publications, it has been observed that the number of publications has increased since 1999, that new proposals have emerged each year, and that proposals based on artificial intelligence techniques have grown in the last few years since 2016. However, there were other approaches, which also have their advantages. For example, a multi-objective technique has shown a good number of supporters in several of the recent publications of the reference report [27, 32], as it has the capacity to address two or more different types of objectives in a single prioritization.

Each approach has potential values, advantages and limitations. The inputs and the type of data play an important role in determining their advantages and limitations. For example, the requirement-based approach uses customer information during obtaining requirements such as entries, to prioritize and generate test cases. The risk-based approach may also use the risk requirement as one of the inputs for executing the prioritization process. This indicates that both may have their own advantages over other approaches in terms of the starting point of TCP execution, as both may start before execution starts and the code is available.

In conclusion, TCP is recognized as an important element in the regression tests, in the current investigations, since it has the ability to increase the effectiveness of testing in terms of failure detection, cost and time.

3 RESEARCH METHOD

The latest in-depth review found [18] contains information on work on prioritisation techniques for test cases until December 2016. The growing importance of software testing requires, among other tasks, prioritizing test cases to make the process more effective and efficient. This has led to a greater interest and research in techniques that achieve this objective, as can be seen in Figure 5 of the aforementioned report. This increase in the number of publications on this subject has served as a motivation for the authors to carry out a study in the three years following this review.

This review attempts to perform a systematic literature review (SLR) on the TCP approaches published in the last three years (2017–2019), according to the methodology defined by Kitchenham [20]. Each step is described in a separate section below.

3.1 Selection of research questions

We have chosen the following research questions:

RQ1 Is the taxonomy defined by Khatibsyarbini et al. still valid for the state of the art in Test Case Prioritisation? New categories may have appeared since then, or the definitions of the categories may need further refinement.

RQ2 Have there been changes in the popularity of the various types of TCP techniques since the original survey? The rise of artificial intelligence in software engineering may have changed the trends in the last few years.

RQ3 How did the new approaches impact the TCP results?

RQ4 How were these new approaches validated?

RQ5 How are the new approaches distributed across application domains?

RQ1 and RQ2 consider changes from the original survey. RQ3 and RQ4 are reused from the original survey. Finally, RQ5 is an original question from this work, given the specialisation of testing techniques across application domains.

3.2 Selection of repositories

We have chosen the following sources to identify primary studies:

- IEEE Xplore
- Scopus
- Google Scholar

The search has been restricted to these three repositories because they include most of the publications of the best known publishers as they are: Springer [36], Elsevier, ACM and Wiley Online Library.

3.3 Search strategy

In the three repositories we have searched the strings *test case prioritization, Test Case Prioritization or test prioritization* and from 2017 to 2019. In addition, in Scopus, we have included the *English* language.

3.4 Selection of inclusion and exclusion criteria

The table 1 describes the inclusion and exclusion criteria that were applied to the primary studies. These were based on those from the original survey.

3.5 Data synthesis and extraction

The three repositories mentioned in Section 3.2 were analyzed:

- IEEE Xplore: 117 papers (41 from 2017, 50 from 2018 and 26 from 2019).
- Scopus: 238 papers (85 from 2017, 95 from 2018 and 58 from 2019).
- Google Scholar: 126 papers (43 from 2017, 54 from 2018 and 29 from 2019).

The results were then gathered, removing duplicates and filtering out those that were outside the scope of the topic, as well as those that were not accessible.

Once this has been done, the results have been classified according to the categories of the referenced review, in order to obtain the trends in these three years, as you can see in the following section.

4 RESULTS AND DISCUSSION

All in all, 320 papers were found in this period about test case prioritization. Across years, 129 correspond to the year 2017, 124 to the year 2018, and finally, 67 were found to be from 2019. Since 2019 is still underway, it follows that there will be fewer publications on the subject. In addition, the original survey goes over 17 years, whereas in the current review, only three years are studied. In any case, the trend in the number of publications on the subject is clearly increasing, since at the end of 2016 there were no more than 20 of them, while in 2017 more than 100 have been found.

4.1 RQ1: Is the taxonomy defined by Khatibsyarbini et al. still valid for the state of the art in Test Case Prioritisation?

In terms of the categories considered, publications have been found for the fourteen categories in table A3 of the reference report. These are based on search algorithms, coverage, faults, requirements, execution history, risks, Bayesian networks, cost effectiveness, topic models, workflows, lexicographical ordering, similarity measures, scope and models.

The new papers motivated the creation of several new categories:

- *Location-based*: Locations and points of interest are correlated by their location proximity via the services. Moreover, many location-based applications use estimated locations and treat similar locations homogeneously. They provide domain-specific heuristics to guide testing techniques to prioritize test case. [42]
- *Machine learning-based*: This technique uses learning, data history, execution times, descriptions in natural language, etc., to prioritize test cases among those most prone to errors. It is used in Continuous Integration environments and in system-level application testing, where the source code is not accessible. [21]
- *Neural network-based*: In order to increase the detection of faults in an application, an important function is defined and algorithms based on neural networks are used. [37]
- *Empirical*: From empirical studies on TCP in a certain domain or with a certain software guidelines are established for the application of TCP techniques. [24]

For some of the new categories there are more than 2 references, as for example in the case of learning machines [21, 22, 39, 43] and neural networks [35, 37, 38]. In addition, the number of proposals that apply more than one technique and take into account various criteria when prioritising test cases are highlighted on Figure 3.

4.2 RQ2: Have there been changes in the popularity of the various types of TCP techniques since the original survey?

In addition, we show the average number of TCP papers per year, by type found (1996–2016, 2017–2019) in Figure 1. The “Search-based" category is still the most prominent, and in these three years the number of publications has increased considerably. “Coverage-based”, “Fault-based” and “History” are also highlighted as in the reference report. However, in the period 2017–2019, “Similarity” stands out in third place, which barely stood out in previous years. Nonetheless, “Risk-based” and “Requirement”, which were in an intermediate position in the previous period, are now very rare. A detailed table of the last three years and by categories can be appreciated in figure 2.

Furthermore, there has been an enormous increase in contributions that use more than one criterion to prioritise test cases. This may indicate that better results are obtained by taking into account a number of factors that can lead to software failures [33]. It can be observed that of the number of publications found in recent years belonging to the multi-criteria category, more than 40% of them improve or optimise an initially proposed technique. As a result, optimization algorithms are applied in [26, 41], or an improvement is made in [25, 33] that reduces the cost of prioritizing test cases.

Likewise, work on prioritisation based on similarity has increased significantly. This may be due to the fact that they allow the use of algorithms and classification techniques that are highly developed and proven, giving good results in real cases. The calculation of the distance between test cases is related to their dissimilarity which allows the detection of test cases with the highest probability of detecting faults [1, 9, 12, 34] .

4.3 RQ3: How did the new approaches impact the TCP results?

The results show that the main TCP approaches used are “search-based”, “coverage-based”, “similarity-based” and “fault-based”. As

Table 1: The inclusion/exclusion criteria.

Inclusion criteria	Exclusion criteria	How applied
English	Other language	Filtering search
Test case prioritization	Other topic	Using these terms
2017–2019	Other dates	Advanced searches
Papers that cite [18]	Cites included in [18]	Searches

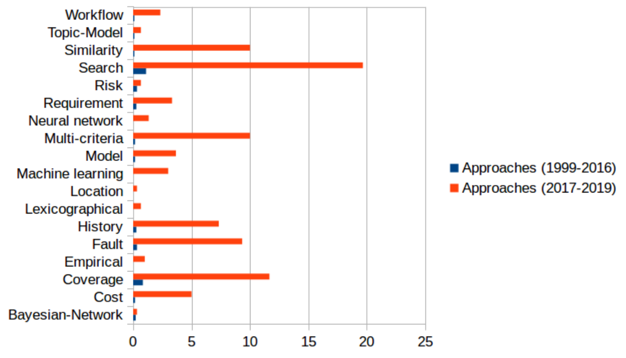
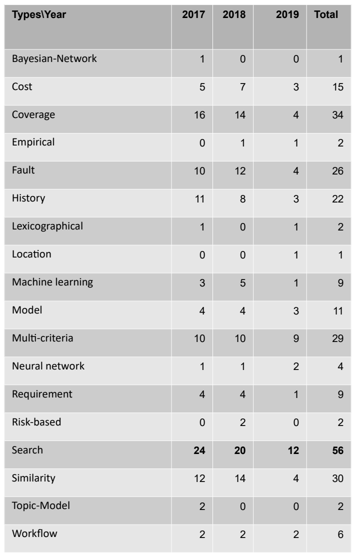


Figure 1: Average number of papers per year, by type (1996–2016, 2017–2019)



Types\Year	2017	2018	2019	Total
Bayesian-Network	1	0	0	1
Cost	5	7	3	15
Coverage	16	14	4	34
Empirical	0	1	1	2
Fault	10	12	4	26
History	11	8	3	22
Lexicographical	1	0	1	2
Location	0	0	1	1
Machine learning	3	5	1	9
Model	4	4	3	11
Multi-criteria	10	10	9	29
Neural network	1	1	2	4
Requirement	4	4	1	9
Risk-based	0	2	0	2
Search	**24**	**20**	**12**	**56**
Similarity	12	14	4	30
Topic-Model	2	0	0	2
Workflow	2	2	2	6

Figure 2: Number of papers per year, by type (2017–2019)

commented before, similarity-based approaches which hardly stood out in the previous report now have a prominent position in terms of the number of publications on the subject.

The focus on cost and multi-criteria has also increased notably. This may be due to the concern to reduce costs in the software, avoiding loss of time, money, etc. In terms of multi-criteria, as mentioned above, it is more efficient to apply techniques that take into account different factors that can cause errors in the software. This combination reduces costs and makes it possible to cover a large part of the code.

Figure 3 shows the number of articles that use one, two or more of the categories listed above to develop their proposal (excluding reviews). In the original survey, some publications are classified into a single type, but in its Table A3 some are put into more than one category.

4.4 RQ4: How were these new approaches validated?

In most cases, they have been validated through their use in real applications with adequate and similar metrics. The following describes the validation results for most of the articles in the new categories.

Wang et al. presented in [42] an empirical evaluation by using one industrial project. Moreover, the experimental results show that the median APFD (Average Percentage Faults Detected) value of Location TCP is 78.57, which is higher than the values of the baseline methods.

In [22], the authors tested the technique on two systems to evaluate the ability of early detection of errors using the APFD

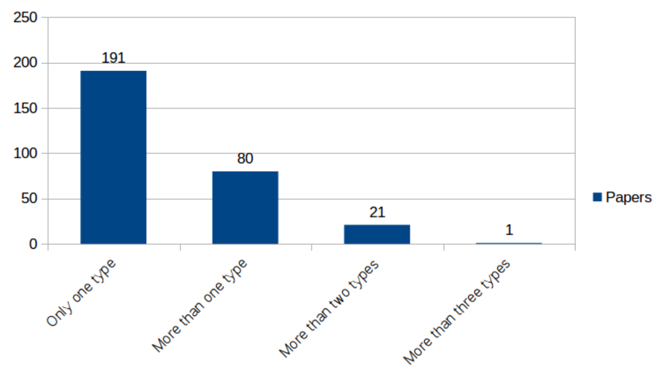


Figure 3: Number of approach types per paper

metric. The results show remarkable improvements over manual and random prioritization by experts.

Wen et al. present in [43] an improvement of the RETECS (REinforced TEst Case Selection) method using the FP-Growth algorithm [1]. They have compared the results with three other prioritisation techniques and the experimental results show that the method is able to incorporate improvements over RETECS by increasing the fault detection ratio. NAPFD (Normalized Average Percentage of Faults Detected) has been used as a metric.

Toure and Badri published a paper [39] on prioritization in unit testing, where the test cases are classes of an object-oriented software. They use two evaluation techniques: CSV (cross-system validation), and LOSOV (leave-one-system-out validation), which merges training sets from different systems. The results indicate all obtained classifiers could help to support unit tests prioritization with more than 70% of accurate predictions.

The LET(learning-to-test) system in [5] uses a learning-to-test approach to accelerate C compiler testing, demonstrating that it reduces the C compiler testing time significantly: between 25% and 50% in more than 36% of the cases.

In the category of neural networks, in [37], the parameters used to measure the results were the error detection ratio and the execution time. A MATLAB simulation was used.

Chaudhury et al. in [4] considered several factors for prioritisation: event type, event interaction, and event coverage. These factors were divided into low, medium, and high ranges. The results achieved a higher APFD rate with a huge number of test cases created to cover the possible events.

4.5 RQ5: How are the new approaches distributed across application domains?

The new approaches are related to service-oriented software, internet of things (IoT), and mobile devices. They generally use artificial intelligence-based techniques. The new categories are described below in relation to the domains in which they are applied. The recent work found on the new category of location-based approaches [42] is described in the domain of mobile devices, taking advantage of the parameters that this type of technology provides.

With regard to machine learning, the domains are: robotics (particularly, a paint robot control system [43]), the automotive sector [22], and unit testing object-oriented software [39]. Lachmann et al. in [22] incorporate TCP as a novelty at the system-level without code access (i.e. industrial software). For black box testing, high level artefacts (i.e. test cases, requirements and faults described in natural language) are used.

Chen et al. apply learning to prioritise test cases for testing compilers [5], whose complexity makes them expensive to test. In the experimental study, they use three mainstream open-source C compilers, namely GCC, LLVM, and Open64 for the x86 64-Linux platform.

In regard to neural networks, Thakur [37] considers 10 projects with 4 software changes in an experimental study carried out on a MATLAB simulation. However, in [4], event driven software testing is the objective domain.

[1]The FP-Growth Algorithm, proposed by Han in [14], is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree)

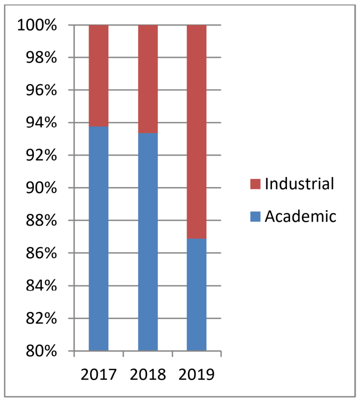


Figure 4: Percentage of approach academic and industrial per year

Liang et al. provide an interesting approach in [23], although it is not among the new categories. This work is an original proposal for continuous integration (CI) environments. It is based on prioritising commits rather than the actual test cases of the software. They have designed an algorithm called CCBP (Continuous Commits-Based Prioritization). It applies to two use cases: Google or Travis-CI.

The table 2 depicts some of the most outstanding domains in the papers. It is worth mentioning the object-oriented programming, software products lines and web applications. As well as industrial projects and industrial automation software. Most software developed today obeys the object-oriented paradigm, and therefore, it is reasonable to test it on its own. In the same way that industrial processes have been modernised and more and more tasks are performed automatically requiring software to support them. In addition, the work on product lines and on web applications stand out, given the high number of activities and transactions that are carried out today through the Internet.

In addition, the papers found have been classified according to whether they are academic or industrial. As can be seen in the figure 4 that shows the percentage of works per year in each of these areas, the percentage of articles in the industrial area grows each year. This could indicate, on the one hand, that in the industrial field, evidence is given increasing importance, and on the other, that it is necessary for industry to collaborate with the academic field, providing real cases to test the proposed techniques.

5 THREATS TO VALIDITY

Similar to previous reviews, the potential threats of this systematic literature review are associated with an incomplete collection of primary studies and no precise data synthesis and derivation.

Table 2: Highlighted domains

Highlighted Domains	number of papers
Automotive	5
Avionics	2
C and Java application	2
Compilers	4
Continuous integration environments	4
Cyber-Physical Systems	4
Defects4J data set	5
Generic	95
Google Dataset	3
GUI software	3
Industrial Projects	7
Industrial Automation Software	8
Java projects	4
Mathematical problems	4
Mobile Software	3
Object-oriented programming	26
Product line	11
Software as a service	4
Web applications	9

In addition to the works classified according to the categories described in the previous section, some reviews have been found. However, many are for specific applications and others include TCP techniques as part of a larger report. For example, the paper by Tzoref [40] deals with advances in combinatorial testing, so that, in addition to the techniques for prioritizing test cases, it examines recent developments in other areas such as fault localization and the evolution of the combinatorial model. Likewise, in the case of work by Ahmad [2], it is specific to the area of event sequences.

6 CONCLUSIONS AND FUTURE WORK

This paper completes the survey from Khatibsyarbini [18] by providing the latest developments in TCP, responding to a growing interest in the literature in this subject. Specifically, we use the taxonomy proposed in that survey including the most important publications from 2017 to the present day (2019).

The results show that TCP is currently of a great deal of interest in the literature. All in all, we found 320 papers in this short period dealing with many different TCP approaches, but the main categories are three: search-based approaches, coverage-based approaches, and similarity-based approaches. A large proportion of the papers combine multiple techniques to optimise the TCP process.

As future work, we would like to determine what the descriptions, strength, and weakness of present prioritization approaches are, how these approaches were applied and affect TCP results, and what processes are involved in TCP. In addition, we would like to use the following criteria to assess the quality of primary studies: the number of cites, the clarity and formality. Finally, it would be useful to classify the papers according to the quality of their evaluation.

ACKNOWLEDGMENTS

Paper partially funded by The Ministry of Economy and Competitiveness (Spain) and the FEDER Fund, under the National Program for Research, Development and Innovation, Societal Challenges Oriented, Project DArDOS TIN2015-65845-C3-3-R, and Project FAME RTI2018-093608-B-C33.

REFERENCES

[1] M. Abu Hasan, M. Abdur Rahman, and M. Saeed Siddik. 2017. Test case prioritization based on dissimilarity clustering using historical data analysis. *Communications in Computer and Information Science* 750 (2017), 269–281. https://doi.org/10.1007/978-981-10-6544-6_25 cited By 2.

[2] J. Ahmad and S. Baharom. 2017. A systematic literature review of the test case prioritization technique for sequence of events. *International Journal of Applied Engineering Research* 12, 7 (2017), 1389–1395. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85018687950&partnerID=40&md5=1ffe00aa61668353f3b13663197f6c27 cited By 3.

[3] M. Azizi and H. Do. 2018. Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization. *Proceedings - 29th IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2018* (2018), 245–251. https://doi.org/10.1109/ISSREW.2018.00014

[4] S. Chaudhury, A. Singhal, and O. P. Sangwan. 2016. Neuro-fuzzy based approach to event driven software testing: A new opportunity. In *2016 1st India International Conference on Information Processing (IICIP)*. 1–5. https://doi.org/10.1109/IICIP.2016.7975349

[5] Junjie Chen, Yanwei Bai, Dan Hao, Yingfei Xiong, Hongyu Zhang, and Bing Xie. 2017. Learning to prioritize test programs for compiler testing. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 700–711.

[6] Omdev Dahiya and Kamna Solanki. 2018. A systematic literature study of regression test case prioritization approaches. *International Journal of Engineering & Technology* 7, 4 (2018), 2184–2191.

[7] Sai Priyatham Dongoor. 2019. Selecting an appropriate Requirements Based Test Case Prioritization Technique.

[8] Sebastian Elbaum, Gregg Rothermel, and John Penix. 2014. 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)*. ACM, New York, NY, USA, 235–245. https://doi.org/10.1145/2635868.2635910

[9] D. Flemström, P. Potena, D. Sundmark, W. Afzal, and M. Bohlin. 2018. Similarity-based prioritization of test case automation. *Software Quality Journal* 26, 4 (2018), 1421–1449. https://doi.org/10.1007/s11219-017-9401-7 cited By 3.

[10] V. Garousi, R. Özkan, and A. Betin-Can. 2018. Multi-objective regression test selection in practice: An empirical study in the defense software industry. *Information and Software Technology* 103 (2018), 40–54. https://doi.org/10.1016/j.infsof.2018.06.007

[11] N. Gupta, V. Yadav, and M. Singh. 2018. Automated regression test case generation for web application: A survey. *Comput. Surveys* 51, 4 (2018). https://doi.org/10.1145/3232520 cited By 0.

[12] A. Haghighatkhah, M. Mäntylä, M. Oivo, and P. Kuvaja. 2018. Test case prioritization using test similarities. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11271 LNCS (2018), 243–259. https://doi.org/10.1007/978-3-030-03673-7_18 cited By 0.

[13] Ines Hajri, Arda Goknil, Fabrizio Pastore, and Lionel C Briand. 2019. Automating Test Case Classification and Prioritization for Use Case-Driven Testing in Product Lines. *arXiv preprint arXiv:1905.11699* (2019).

[14] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *ACM sigmod record*, Vol. 29. ACM, 1–12.

[15] R. Huang, Q. Zhang, T.Y. Chen, J. Hamlyn-Harris, D. Towey, and J. Chen. 2018. An Empirical Comparison of Fixed-Strength and Mixed-Strength for Interaction Coverage Based Prioritization. *IEEE Access* 6 (2018), 68350–68372. https://doi.org/10.1109/ACCESS.2018.2879638 cited By 0.

[16] Rubing Huang, Weiwen Zong, Tsong Yueh Chen, Dave Towey, Yunan Zhou, and Jinfu Chen. 2018. Prioritising abstract test cases: an empirical study. *IET Software* (2018).

[17] J. Jia and X. Liu. 2018. Improving Systematic Literature Review Based on Text Similarity Analysis. *Journal of Physics: Conference Series* 1069, 1 (2018). https://doi.org/10.1088/1742-6596/1069/1/012059

[18] Muhammad Khatibsyarbini, Mohd Adham Isa, Dayang N.A. Jawawi, and Rooster Tumeng. 2018. Test case prioritization approaches in regression testing: A systematic literature review. *Information and Software Technology* 93 (2018), 74 – 93. https://doi.org/10.1016/j.infsof.2017.08.014

[19] Barbara Kitchenham. 2004. Procedures for performing systematic reviews. *Keele, UK, Keele University* 33, 2004 (2004), 1–26.

[20] Barbara Kitchenham, O Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology* 51, 1 (2009), 7–15.

[21] Vinit Kudva. 2018. *Fault Driven Supervised Tie Breaking for Test Case Prioritization.* Ph.D. Dissertation. University of Waterloo.

[22] R. Lachmann, M. Nieke, C. Seidl, I. Schaefer, and S. Schulze. 2017. System-level test case prioritization using machine learning. *Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016* (2017), 361–368. https://doi.org/10.1109/ICMLA.2016.163 cited By 3.

[23] J. Liang, S. Elbaum, and G. Rothermel. 2018. Redefining Prioritization: Continuous Prioritization for Continuous Integration. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. 688–698. https://doi.org/10.1145/3180155.3180213

[24] Qi Luo, Kevin Moran, Lingming Zhang, and Denys Poshyvanyk. 2018. How do static and dynamic test case prioritization techniques perform on modern software systems? An extensive study on GitHub projects. *arXiv preprint arXiv:1806.09774* (2018).

[25] M.H. Mahmood and M.S. Hosain. 2018. Improving test case prioritization based on practical priority factors. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS* 2017-November (2018), 899–902. https://doi.org/10.1109/ICSESS.2017.8343055 cited By 1.

[26] B. Manaswini and A. Rama Mohan Reddy. 2019. A shuffled frog leap algorithm based test case prioritization technique to perform regression testing. *International Journal of Engineering and Advanced Technology* 8, 5 (2019), 671–674. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85069924907&partnerID=40&md5=1a63e2ab19533064658633f048ff66f2 cited By 0.

[27] A. Marchetto, M. M. Islam, W. Asghar, A. Susi, and G. Scanniello. 2016. A Multi-Objective Technique to Prioritize Test Cases. *IEEE Transactions on Software Engineering* 42, 10 (Oct 2016), 918–940. https://doi.org/10.1109/TSE.2015.2510633

[28] Nasir Mehmood Minhas, Kai Petersen, Jürgen Börstler, and Krzysztof Wnuk. 2018. Regression testing for large-scale embedded software development–Exploring the state of practice. (2018).

[29] Deepti Bala Mishra, Namita Panda, Rajashree Mishra, and Arup Abhinna Acharya. 2018. Total fault exposing potential based test case prioritization using genetic algorithm. *International Journal of Information Technology* (2018), 1–5.

[30] Myers, G.J., Sandler, C., Badgett, T., and Thomas, T. M. 2004. *The Art of Software Testing, 2nd ed.* Wiley - Interscience.

[31] Gayatri Nayak and Mitrabinda Ray. 2018. A Guided Approach to Prioritize Object-Oriented Test Suites Using MC/DC Testing. In *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*. IEEE, 186–191.

[32] José A. Parejo, Ana B. Sánchez, Sergio Segura, Antonio Ruiz-Cortés, Roberto E. Lopez-Herrejon, and Alexander Egyed. 2016. Multi-objective test case prioritization in highly configurable systems: A case study. *Journal of Systems and Software* 122 (2016), 287 – 310. https://doi.org/10.1016/j.jss.2016.09.045

[33] A.B. Sánchez and S. Segura. 2017. SmarTest: A test case prioritization tool for drupal. *ACM International Conference Proceeding Series* 2 (2017), 9–12. https://doi.org/10.1145/3109729.3109757 cited By 0.

[34] A.B. Sánchez, S. Segura, J.A. Parejo, and A. Ruiz-Cortés. 2017. Variability testing in the wild: the Drupal case study. *Software and Systems Modeling* 16, 1 (2017), 173–194. https://doi.org/10.1007/s10270-015-0459-z cited By 18.

[35] H. Spieker, A. Gotlieb, D. Marijan, and M. Mossige. 2017. Reinforcement learning for automatic test case prioritization and selection in continuous integration. *ISSTA 2017 - Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis* (2017), 12–22. https://doi.org/10.1145/3092703.3092709 cited By 12.

[36] SpringerLNCS 2019. Information on Abstracting and Indexing. Retrieved July 29, 2019 from https://www.springer.com/gp/computer-science/lncs/information-on-abstracting-and-indexing/799288

[37] Akshit Thakur and Gitika Sharma. 2018. Neural Network Based Test Case Prioritization in Software Engineering. In *International Conference on Advanced Informatics for Computing Research*. Springer, 334–345.

[38] A. Thakur and G. Sharma. 2019. Neural Network Based Test Case Prioritization in Software Engineering. *Communications in Computer and Information Science* 956 (2019), 334–345. https://doi.org/10.1007/978-981-13-3143-5_28 cited By 0.

[39] F. Toure and M. Badri. 2018. Prioritizing unit testing effort using software metrics and machine learning classifiers. *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE* 2018-July (2018), 653–658. https://doi.org/10.18293/SEKE2018-146 cited By 0.

[40] R. Tzoref-Brill. 2019. Advances in Combinatorial Testing. *Advances in Computers* 112 (2019), 79–134. https://doi.org/10.1016/bs.adcom.2017.12.002 cited By 0.

[41] R. Uma Maheswari and D. Jeya Mala. 2017. Heuristic-based time-aware multi-criteria test case prioritisation technique. *International Journal of Information Systems and Change Management* 9, 4 (2017), 315–333. https://doi.org/10.1504/IJISCM.2017.091275 cited By 0.

[42] X. Wang, H. Zeng, H. Gao, H. Miao, and W. Lin. 2019. Location-Based Test Case Prioritization for Software Embedded in Mobile Devices Using the Law of Gravitation. *Mobile Information Systems* 2019 (2019). https://doi.org/10.1155/2019/9083956 cited By 0.

[43] Wen Wen, Zhongju Yuan, and Yuyu Yuan. 2018. Improving RETECS method using FP-Growth in continuous integration. In *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*. IEEE, 636–639.

[44] Uğur Yılmaz. 2019. *A METHOD FOR SELECTING REGRESSION TEST CASES BASED ON SOFTWARE CHANGES AND SOFTWARE FAULTS*. Master's thesis. Fen Bilimleri Enstitüsü.

[45] S. Yoo and M. Harman. 2012. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability* 22, 2 (2012), 67 – 120. http://search.ebscohost.com.bibezproxy.uca.es:2048/login.aspx?direct=true&db=egs&AN=71933963&site=ehost-live

[46] Chuan Yue. 2018. A projection-based approach to software quality evaluation from the usersâĂŹ perspectives. *International Journal of Machine Learning and Cybernetics* (2018), 1–13.