ISSN 1870-4069

Optimization of Combinatorial Software Testing: A Systematic Literature Review

Puxka Acosta-Domínguez, Angel J. Sánchez-García, Candy Obdulia Sosa-Jiménez

Universidad Veracruzana,
Facultad de Estadística e Informática,
Mexico

puxka.acodom@gmail.com, {angesanchez, cansosa}@uv.mx

Abstract. One of the reasons why Industry 4.0 is a reality is the development of quality Software that allows the interconnectivity and management of the devices we use. To ensure quality in the software, it is necessary that it be released with the least number of failures. For this, it is necessary that the software tests cover all possible paths extensively and it is for this reason that approaches such as combinatorial tests arise. This type of test seeks to generate test cases for the detection of failures. This Systematic Literature Review (SLR) compiles the optimization techniques, approaches, areas of opportunity and application of combinatorial tests. In this SLR, 82 primary studies were identified, based on the Kitchenham and Charters guide, where it is observed that the most frequent application of this type of test is the web, although most of the studies only propose improvements without any specific domain. Also, the most widely used approaches are IPOG and its variants. Finally, the most mentioned opportunity area was the improvement of results and algorithms.

Palabras clave: Combinatorial testing, optimization, systematic literature review, quality software.

1 Introduction

In recent years, the need for high-quality software has increased. This is due to the fact that all current devices and those generated in Industry 4.0 require Software that is less prone to failures, and that quality does not impact human and economic resources. This quality is ensured when the Software is thoroughly tested before it is released.

The activities carried out in the development of a software project are divided into processes, such as requirements, design, construction, testing, configuration management, among others [1], being the testing stage the interest of this document. Since depending on the system and the defect sought, it is the type of testing technique that will be used [2]. With the interest in combinatorial tests, this paper presents a Systematic Literature Review (SLR) focused on the categorization of current research.

The aim is to find techniques to design combinatorial tests that can be used for future research areas. This work is aimed at researchers to find niches of opportunity in this area. Other beneficiaries will be students and developers, especially those in the testing

Table 1. Research questions.

Quesition	**Motivation**
Q1.- What are the applications of combinatorial test?	To know where the combinatorial tests have currently been applied.
Q2.- Which optimization techniques have been proposed to generate combinatorial test cases?	To identify proposals for the Artificial Intelligence optimization area in the generation of combinatorial test cases.
Q3.- What areas of opportunity are there for the research community?	To present proposals for areas of research opportunity on the subject of combinatorial test design.

Table 2. Identified keywords and synonyms.

Concept	**Synonyms**
Combinatorial testing	Combinatorial test, combinatorial interaction testing, pairwise testing
Automation	Automated, automatized
Optimization	Optimize, optimized
Algorithm	Approach, strategy, method

area. The first will get an insight into a topic that has been little covered, and developers in the industry will get a starting or improvement point for their current testing. The work will use a systematic method to extract and analyze the works on the topic.

The aim of this Systematic Literature Review is to present the state of the use of Artificial Intelligence in the design of combinatorial tests to ensure the quality of the software. This is to discuss the most promising approaches and, in the future, generate lines of research in the area.

This paper is organized as follows: Section II presents the background and related work of reviews on the subject. Section III exposes the method followed to carry out the SLR. Section IV describes the final results. Finally, Section V draws conclusions.

2 Background and Related Work

Combinatorial tests are a method that can detect defects that other types of tests cannot (for example, random tests or equivalence partition tests), since the defect can be found in the combination of parameters. In addition, they reduce the number of tests, therefore reducing costs [3]. These tests are of the black box type, therefore, to carry them out, test cases are created that receive inputs to a functionality and as a result provide an output. Combinatorial tests can be from two to T variables. Combinatorial tests have also been called as combinatorial tests in pairs, which are those of two variables, and tests of combinatorial interaction.

All combinations can be represented by a covering array (CA). The CA is a mathematical object in which all combinations of a parameter are covered at least once [4]. These can be created with the help of tools, which change depending on the algorithm on which they are based. There are different types of algorithms, among those based on Artificial Intelligence (AI) are optimization algorithms. AI has sought to reduce the human error that is always present in any system, therefore, it is not

Table 3. Inclusion Criteria.

IC	Description
1	Studies published between the years 2015 to 2021
2	Studies written in English
3	The title and / or abstract give indications that at least one research question will be answered
4	The full text answers at least one research question

Table 4. Exclusion Criteria.

EC	Description
1	It is a summary, workshop, opinion piece, presentation, book or technical report.
2	Do not have access to the full text
3	It is a duplicate research
4	The full text answers at least one research question

surprising that this discipline collaborates with Software Engineering (SE). According to a study [5], the use of AI algorithms in the SE testing phase has been investigated since 1970 and mostly in black box testing, of which combinatorial testing is a part.

The oldest review on the subject found dates from 2014 [6] and it took the topic of combinatorial testing as part of Software Product Line testing strategies. Other review [7], collected the algorithms and tools from 1991 to 2014, but is not strict in following a method.

The only Systematic Mapping found [8] has as main focus combinatorial tests with application in SPL. A Systematic Literature Review from 2017 [9] talks about combinatorial tests in general, mentioning tools and limitations, but it is the one that has its method with less detail of all the reviews.

The SLR found [10] describes the combinatorial tests that have handling of restrictions, which are also mentioned in the present work, but not limited to these.

3 Research Method

To carry out this review, the method proposed in [11] by Kitchenham and Charters was performed, which is a guide used in Software Engineering to carry out Systematic Literature Reviews (SLR). This section describes the steps defined in [11].

3.1 Research Questions

Table 1 shows the research questions with their motivation, to guide this process.

3.2 Search Strategy and Data Sources

The automated search strategy was used to collect the information. For this, keywords were identified, which were taken from the research questions and similar terms, as shown in Table 2.

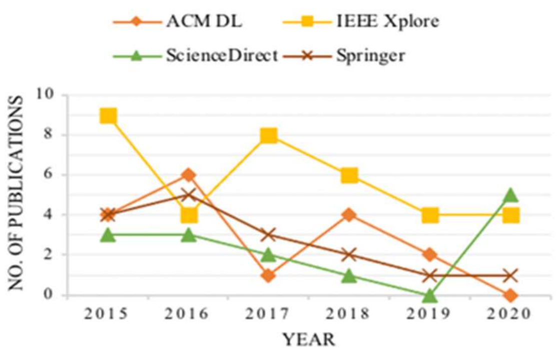

Fig. 1. Primary studies by year of publication.

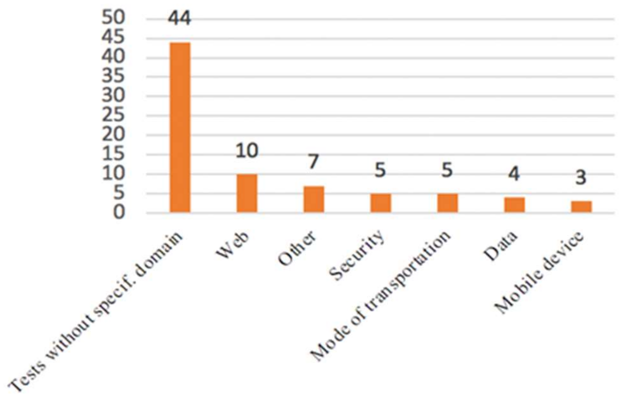

Fig. 2. Number of primary studies per application of combinatorial tests.

With these terms, 5 search strings were created and they were validated with the precision and recall criteria, which have been used in Systematic Reviews of Software Engineering [12]. The results of these metrics for each of the 5 search strings can be found in Appendix A[1].

The selection of sources was based on [11] and [12], two guides for conducting systematic reviews of Software Engineering. Following these, the following repositories were selected: SpringerLink, IEEE Xplore, ScienceDirect and ACM Digital Library.

3.3 Selection Criteria

For the selection of primary studies, inclusion and exclusion criteria were established, which can be seen in Table 3 and 4.

[1] Appendix A. Search String metrics: https://drive.google.com/file/d/1B_7ZuB0EFYNzbYmtQwAZt1SWhGk0ZA9R/view?usp=sharing

Table 5. Application of inclusion and exclusion criteria by stage.

Database	First Results	Stage 1	Stage 2	Stage 3
ACM Digital Library	400	228	30	17
IEEE Xplore	360	177	44	35
ScienceDirect	420	183	15	14
SpringerLink	433	168	22	16
Total	1613	756	111	82

Table 6. Quality assessment criteria.

NC	Criteria
1	Is the study research-based (not just a report based on “lessons learned” or opinion)?
2	Is there an explanation of the objectives of the research?
3	Is there a description of the context in which the research was carried out?
4	Does the research design speak to the research objectives?
5	Did the sample of data used have a description?
6	Was the information collected in a way that spoke to the research problem?
7	Was the information analysis rigorous?
8	Is there an explanation for the findings?
9	Is the study of value for research or practice?

3.4 Selection Procedure

The selection process is made up of the following stages:

– Stage 1. Primary studies are filtered according to IC1 and IC2 and studies are removed according to EC1.
– Stage 2. The primary studies are filtered according to the IC3 and the primary studies are removed according to the IC2 and EC3.
– Stage 3. Primary studies are filtered according to IC4.

First, the automated search was executed in the four repositories, being 1613 the works found, as can be seen in Table 5. After that, the inclusion and exclusion criteria were applied stage by stage, reducing the studies in SpringerLink by 95.75%, in IEEE Xplore by 90.28%, in ScienceDirect by 96.67% and in ACM Digital Library by 96.08%.

3.5 Quality Assessment

The criteria to assess the quality of the primary studies were obtained from a systematic review used and accepted in the Software Engineering community [13]. The criteria are in the form of a questionary. The selected criteria are shown in Table 6. Each question is answered with “yes” or “no” (rated as 1 or 0 respectively). So that each study can

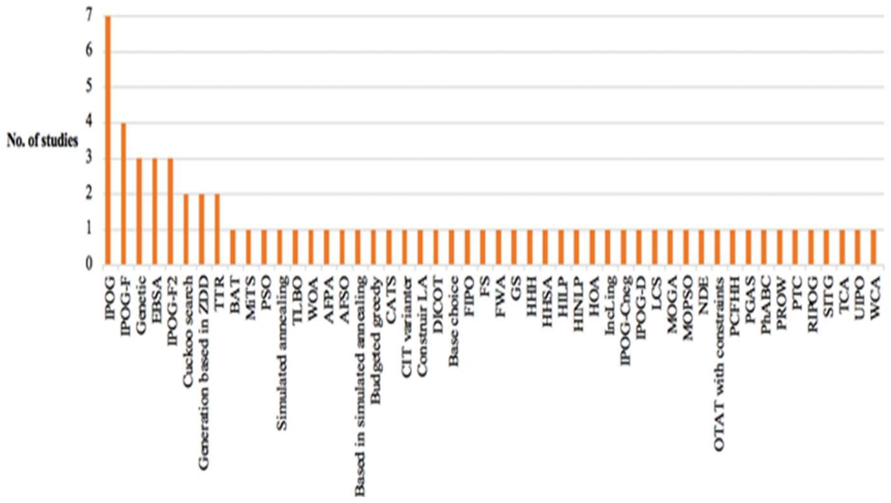

Fig. 3. Algorithms most used in combinatorial testing.

have a final score from 0 (which means very poor) to 9 (which means very good). The quality score of the selected works can be seen in a freely accessible spreadsheet [14].

4 Results

In Fig. 1 the primary studies that passed the selection process are shown. It was found that three digital libraries have been a decrease in the combinatorial test research. Each of the research questions is answered in summary below.

4.1 Q1: What are the Applications of Combinatorial Test?

As can be seen in Fig. 2, seven main applications of combinatorial tests were identified in the last 6 years. All studies whose approach to performing combinatorial tests did not have a specific application domain were pooled. In the *"Other"* category, the domains that were only present in a primary study were grouped.

4.2 Q2: Which Optimization Techniques Have Been Proposed to Generate Combinatorial Test Cases?

From all the studies, 54 papers answered this question, which represent 66% of the total number of studies selected. These mention around 50 optimization algorithms, all with a varied level of detail. Even with this variation, we organize them into the original algorithms, those that were based on an existing one to improve it as it is shown in Table 7 and those that use another algorithm as part of their own process, as can be seen

Table 7. Algorithms based in other optimization algorithms.

Base Algorithm	**Proposed Algorithm**
ABC	PhABC
AETG System	Prow
BSA	EBSA
Cuckoo Search	LCS
D-construction	RIPOG
FPA	AFPA
Genetic	GS, HOA, PGAS
Greedy approach	Budgeted greedy, DICOT, FIPO, HILP, HINLP, IncLing, IPOG, IPOG-F, IPOG-F2, OTAT, PROW, PTC, TTR
IPO	IPOG, IPOG-D, IPOG-F, IPOG-F2, UIPO
IPOG	RIPOG
IPOG-C	IPOG-CNeg
IPOG-D	RIPOG
OTAT	EBSA, FS, FWA, OTAT with constraint, PROW
PSO	MOPSO, SITG
Simulated annealing	AFSO
Tabu search	CATS

Table 8. Algorithms used with another optimization algorithm.

Base Algorithm	**Proposed Algorithm**
Ant Colony	UIPO
ExtendCA	AFSO
FPA	FIPO, PCFHH
Greedy Search	PCFHH

in Table 8. The detail of each algorithm per paper and its reference can be seen in Appendix B[2].

As can be seen in Fig. 3, the most used algorithm was In-Parameter-Order-General (IPOG) since ACTS, the most used tool, uses this algorithm and variations. The most used algorithm from the original algorithms is the Genetic Algorithm (GA), the second is Cuckoo search and the others were only used once.

Most of the original algorithms are bioinspired. Cuckoo search, Bat algorithm (BAT) and Whale Optimization Algorithm (WOA) are algorithms inspired by animal behaviors, Simulated Annealing in the process of heat treatment of metal, Teaching Learning Based Optimization (TLBO) in school teaching and Particle Swarm Optimization (PSO) in social behavior.

[2] Appendix B. Algorithms by paper: https://drive.google.com/file/d/19qJ8PsWntCOToq73-SIryjuMq-RF6PL3/view?usp=sharing

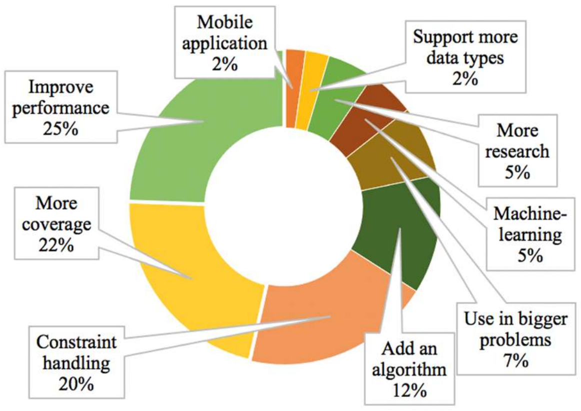

Fig. 4. Percentage of opportunity areas mentioned in the studies.

4.3 Q3.- What Areas of Opportunity are There for the Research Community?

The 9 areas of opportunity mentioned in the studies are shown in Fig. 4. One of the most mentioned was improving the performance of algorithms, although this type of test is known to give optimal results in its performance. It is wanted to decrease the size of the test set [15] or improve search efficiency in a large set of test cases [16]. The possibility exists of achieving this by parallelizing the algorithm [17]. Later, the majority mentioned the increase in coverage of variables. This in combinatorial tests in pairs [18], 3-variables [19] and one of 6-variables [20].

Another interest is adding constraint handling. Many of the tools of the last years do not implement any method for this [21]. Therefore, it is the intention to apply them in well-known algorithms [22] or in algorithms that have only just been introduced to combinatorial tests [23]. It also seeks to complement or combine an algorithm with some other to improve the results [24], but they want to incorporate more to refine the algorithm.

Also, testing the algorithm in big problems is mentioned. The goal would be to scale to very large problems and find smaller sets of tests than in your current results [25]. The sixth most mentioned is to integrate machine-learning to the algorithm. This could help test generation by learning from previously generated test sets [22].

5 Conclusions and Future Work

This work took an approach that few works have taken and, unlike those that have, the method proposed in [11] was detailed and followed. With this, the reproducibility of the results is possible. In addition, the results of the extraction and quality evaluation are online and available to all public. As a result, the algorithms that have been applied to generate the combinations of test cases have been compiled. These algorithms are the basis for combinatorial tests, which are applied in a wide variety of fields such as web, security, mobile devices and others, which shows that it is an active area, current, but above all, a useful area to ensure the quality of software of any kind.

In this study, it was observed that the most widely used approaches are IPOG and its variants, bioinspired approaches and classic approaches such as Simulated Annealing and Greedy search. Therefore, it can be seen that although there is a growing interest in bioinspired algorithms, some other metaheuristics have contributed much knowledge in this area of Software Engineering.

Finally, the most mentioned opportunity area was the improvement of results and algorithms. This is important since based on the proposed approaches, you can explore and experiment with others to compare the results. Nevertheless, the greater coverage and handling of restrictions are also highlighted. Therefore, it is seen that there is still a lot of research to be done in this field and with the help of new optimization strategies, it is sought to ensure the quality of the software, which is fundamental in Industry 4.0.

As future work it is proposed to explore the comparison of bioinspired algorithms with other metaheuristics (such as simulated annealing). Also explore local search methods to get better coverage in test cases.

References

1. Bourque, P., Dupuis, R., Abran, A., Moore, J. W., Tripp, L.: Guide to the software engineering body of knowledge. IEEE Software, IEEE Computer Society, vol. 16, no. 6, pp. 35–44 (1999) doi: 10.1109/52.805471
2. Wu, H., Nie, C., Petke, J., Jia, Y., Harman, M.: An empirical comparison of combinatorial testing, random testing, and adaptive random testing. IEEE Trans. Softw. Eng., vol. 46, no. 3, pp. 302–320 (2020) doi: 10.1109/TSE.2018.2852744
3. Kuhn, R., Kacker, R., Lei, Y., Hunter, J.: Combinatorial software testing. Computer (Long. Beach. Calif), vol. 42, no. 8, pp. 94–96 (2009) doi: 10.1109/ MC.2009.253
4. Sloane, N. J. A.: Covering arrays and intersecting codes. J. Comb. Des., vol. 1, no. 1, pp. 51–63 (1993) doi: 10.1002/jcd.3180010106
5. Lima, R., Da Cruz, A. M. R., Ribeiro, J.: Artificial intelligence applied to software testing: A literature review. In: Iberian conference on information systems and technologies. CISTI, vol. 2020, no. June (2020). doi: 10.23919/CISTI49556. 2020.9141124
6. Machado, I. D. C., McGregor, J. D., Cavalcanti, Y. C., De Almeida, E. S.: On strategies for testing software product lines: A systematic literature review. Inf. Softw. Technol., vol. 56, no. 10, pp. 1183–1199 (2014) doi: 10.1016/j.infsof. 2014.04.002
7. Khalsa, S. K., Labiche, Y.: An orchestrated survey of available algorithms and tools for combinatorial testing. In: Proceedings International Symposium on Software Reliability Engineering, ISSRE, pp. 323–334 (2014) doi: 10.1109/ISS RE.2014.15
8. Lopez-Herrejon, R. E., Fischer, S., Ramler, R., Egyed, A.: A first systematic mapping study on combinatorial interaction testing for software product lines (2015) doi: 10.1109/ICSTW.2015.7107435
9. Abdullah, A., Hassan, R., Shah Z. A.: A Systematic literature review of combinatorial testing. Int. J. Advance Soft Compu. Appl, vol. 9, no. 2 (2017)
10. Ahmed, B. S., Zamli, K. Z., Afzal, W., Bures, M.: Constrained interaction testing: A systematic literature study. IEEE Access, vol. 5, pp. 25706–25730 (2017) doi: 10.1109/ACCESS.2017.2771562

11. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Department of Computer Science University of Durham, Durham, UK, Tech Rep (2007)
12. Dieste, O., Grimán, A., Juristo, N.: Developing search strategies for detecting relevant experiments. Empirical Software Engineering, vol. 14, no. 5, pp. 513–539 (2009) doi: 10.1007/s10664-008-9091-7
13. Dyba, T., Dingsøyr, T.: Empirical studies of agile software development: A systematic review. Information and software technology, vol. 50, no. 9-10, pp. 833–859 (2008) doi: 10.1016/j.infsof.2008.01.006
14. Acosta, P.: Estudios primarios. Estrategias para diseñar pruebas combinatorias (2020)
15. Balera, J. M., de Santiago, V. A.: A controlled experiment for combinatorial testing. In: SAST: Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing, pp. 1–10 (2016) doi: 10.1145/2993288.2993289
16. Wolde, B. G., Boltana A. S.: Combinatorial testing approach for cloud mobility service. In: AICCC 2019: Proceedings of the 2019 2nd Artificial Intelligence and Cloud Computing Conference, pp. 6–13 (2019) doi: 10.1145/3375959.3375967
17. Al-Hajjaji, M., Krieter, S., Thüm, T., Lochau, M., Saake, G.: IncLing: Efficient product-line testing using incremental pairwise sampling. ACM SIGPLAN Not., vol. 52, no. 3, pp. 144–155 (2016) doi: 10.1145/2993236.2993253
18. Alazzawi, A. K., Rais, H. M., Basri, S., Alsariera, Y. A.: PhABC: A hybrid artificial bee colony strategy for pairwise test suite generation with constraints support. In: IEEE Student Conference on Research and Development, SCOReD´19. pp. 106–111 (2019) doi: 10.1109/SCORED.2019.8896324
19. Kampel, L., Simos, D. E.: Set-based algorithms for combinatorial test set generation. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9976, pp. 231–240 (2016)
20. Ahmed, B. S.: Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. Engineering Science and Technology, an International Journal, vol. 19, no. 2, pp. 737–753 (2016) doi: 10.1016/j.jestch.2015.11.006
21. Petke, J.: Constraints: The future of combinatorial interaction testing. In: Proceedings - 8th International Workshop on Search-Based Software Testing, SBST´15, pp. 17–18 (2015) doi: 10.1109/SBST.2015.11
22. Li, N., Lei, Y., Khan, H. R., Liu, J., Guo, Y.: Applying combinatorial test data generation to big data applications. In: ASE 2016. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, pp. 637–647 (2016) doi: 10.1145/2970276.2970325
23. Ahmed, B. S.: Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. Engineering Science and Technology, an International Journal, vol. 19, no. 2, pp. 737–753 (2016) doi: 10.1016/j.jestch.2015.11.006
24. Ericsson, S., Enoiu, E.: Combinatorial modeling and test case generation for industrial control software using ACTS. In: Proceedings IEEE 18th International Conference on Software Quality, Reliability, and Security, QRS´18. pp. 414–425 (2018) doi: 10.1109/QRS.2018.00055
25. Qi, R. Z., Wang, Z. J., Li, S. Y.: A parallel genetic algorithm based on spark for pairwise test suite generation. J. Comput. Sci. Technol., vol. 31, no. 2, pp. 417–427 (2016) doi: 10.1007/s11390-016-1635-5