Journal of Information Systems Engineering and Management
2025, 10(5s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Rule-Based Approach for Identifying Failure Inducing Combinations in Combinatorial Testing

Rekha Jayaram1*, Krishnan Rangarajan2

[1]Department of Information Science and Engineering, Dayananda Sagar College of Engineering, Bangalore, Visvesvaraya Technological University, Belagavi – 590018, Karnataka, India.

[2]Department of Computer Science and Engineering, Dayananda Sagar College of Engineering, Bangalore, Visvesvaraya Technological University, Belagavi – 590018, Karnataka, India.

*Corresponding author: Rekha Jayaram

ARTICLE INFO

Received: 11Oct 2024
Revised: 12Dec 2024
Accepted:24Dec 2024

ABSTRACT

In complex software systems, identification of parameter combinations that lead to failures is important for effective debugging and robust software quality assurance. Combinatorial testing (CT) can greatly reduce the number of test cases (TC) used for testing a complex system by generating TCs based on various parameter combinations, while still maintaining high fault-detection capabilities. In this paper, a rule-based approach is presented that will aid in identifying Failure Inducing Combinations (FIC) of parameters that caused a fault in CT generated TCs. The approach uses heuristics to identify rules to methodically split parameter combinations into more likely or less probable combinations causing failure, thus reducing the set of failure sources. This approach is tested on two different case studies i) Three factor authentication system and ii) Existing literature-based input. The results obtained show significant accuracy and time saving for program debugging, thereby indicating applicability of the approach to real-world problems. The approach was found to be successful in identifying the pair-wise combinations of parameters that are likely to cause failure.

Keywords: Failure Inducing Combinations, Combinatorial Testing, Rule-based approach, Fault Localization, Fault Characterization.

INTRODUCTION

CT is a technique for software testing that systematically generates test cases by using combinations of parameters. The test cases would have several parameters intermixed using different techniques so as to consider all possible interactions among these parameters. CT significantly reduces the number of TCs from exhaustive testing while maintaining high fault detection capability. This makes it particularly good at picking out failure cases caused due to the interaction of parameters, which is a common phenomenon for complex software systems. Since CT systematically tests all possible parameter interactions, it helps to detect faults not visible when the parameters are tested in isolation [1] [2]. A systematic review of the various CT approaches based on various criteria such as CT generation strategy, supported interactions, support constraints between parameters and others can be found in [3]. The process of determining the exact parameter combinations i.e. FICs that lead to a failure in a SUT is called Fault Localization (FL). Consequently, a failure in a test case may not necessarily be fully attributed to all parameters involved in that test case. What FL techniques attempt to do is trace the actual parameters or the combinations thereof that induce the fault. Debugging then becomes an important exercise as it identifies a smaller subset of causes for the failure, thereby increasing manageability in the correction of underlying issues [4] [5].

There are several reasons why identification of the FICs is critical such as:

- Efficient debugging: By identifying exact combinations of parameters that are failure-inducing, developers can concentrate their debugging efforts efficiently. This decreases both the time and resources involved in the identification and fixing of bugs and accelerates resolution and efficient maintenance of the software [6] [7].
- Improved software quality: Precise understanding of the occurring failures boosts the robustness and reliability of software; therefore, quality is improved. Root causes of failures are eliminated, so their similar appearance in the future is reduced [2] [8].
- Cost reduction: Faults, if detected and removed at an early stage, then the cost of developing and maintaining software will be reduced considerably. Later found faults are expensive; hence, to keep

costs lower, the use of combinatorial testing techniques with effective fault localization techniques can be better performed [9] [10].

- Support for complex systems: Modern software systems are highly complex, consisting of a great deal of interrelated components and parameters. Traditional testing methods may not provide enough support for such complexity. Effective FL combined with CT provides a structured approach to manage such complexity and to achieve thorough testing of all significant interactions [11] [12].

Integration of CT and FL in a software development team can effectively, efficiently, and economically test for the production of high-quality software products. A rule-based approach is clear, effective, and systematic in the context of the same [13] [14]. This approach uses some predetermined set of rules or conditions for making decisions.

2. RELATED WORK

CT works on effectively combining parameters so as to reduce the TCs by efficiently combining the parameters of the SUT [15]. Upon executing the CT test suite, some TCs may fail due to the combination of parameters. These parameter combinations need to be analysed to identify the FICs. Several approaches to identifying FICs have been proposed till now. These approaches can be broadly classified into Non-adaptive and Adaptive approaches - based on generation and execution of additional TCs [16] [17]. Research work reported on these approaches from 2009 to 2024 are summarised in this section. The survey was done with focus on FL in CT.

Adaptive approaches take CT results as inputs and generate additional TCs to help identify the FICs by using different techniques for these TCs generation. For generating additional TCs, some techniques use a single failed TC as input and additional TCs are generated by modifying the values of the parameters present in the failed TC. The techniques used here are OFOT [18] and the techniques based on Delta debugging [19]. Some techniques generate additional TCs by taking multiple failed TCs as input as discussed in [9] [20] [21] [22]. Other techniques take all failed TCs as input to generate additional TCs as discussed in [2] [7] [23] [24] [25] [26]. Some of the disadvantages of this approach is that a single failed test case or only a few of them would not be sufficient in identifying the additional TCs. [27] proposes a MIXTGTE technique in which the additional TCs are generated using the results of the previous tests executed. [28] proposes a greedy algorithm based Adaptive approach for identifying the FICs.

Non-adaptive approaches take CT results as inputs and analyze the same to detect potential faulty interactions. These approaches mainly generate a Locating array to help design a Covering array. The approach based on SAT solving proposed by Zhang et.al [29], approach based on Error locating array as proposed in [30], partial covering array proposed by [31], generation (1,2) Locating array as given in [32] and Constrained Detecting arrays proposed in [33] are examples of this type of approach. Some of the disadvantages of this approach is the need for information about the failure inducing schemas to generate Locating arrays, assumptions on this number of faulty interactions and the increase in the size of the Locating array as compared to the original covering array thus contributing to increase in the execution cost. An effective use of covering arrays by utilising the prior knowledge of the SUT is shown in [34].

Some more work done on Fault Localization in Combinatorial testing is discussed here.

Jin, et al., [35] proposes Constraint Detection Arrays (CDAs) as an extension of Detecting Arrays (DAs) which can be applied in Combinatorial Interaction Testing environments where systems have constraints on test parameters. They are particularly useful for identifying and localizing faults in complex systems where traditional detecting arrays may fail due to presence of constraints. The technique proposed here assigns weights to TCs and are then executed based on the priority given based on the weights. It also uses a greedy technique to help execute the TCs.

Bonn, et al., [36] introduces COFFEe (COmbinatorial test and Fault characterization FramEwork), which integrates all activities necessary for combinatorial testing such as test input generation, TCs execution and also fault characterization. It can be used in any software testing environment requiring combinatorial testing and fault localization. It is particularly useful for systems that have complex input parameters and constraints, ensuring comprehensive test coverage and efficient FL.

Kampel, et al., [37] introduces CT based FL approaches (CT-FLA) to enhance explainability in AI systems. The method uses CT to identify failure inducing interactions and applying these learnings to explain AI decisions. The core idea is to leverage combinatorial interaction testing (CIT) to map the AI system's input to its outputs and identify feature combinations that influence these outputs.

Niu, et al., [20] introduces an interleaving approach that combines test case generation and failure-inducing interaction identification. This framework integrates these two stages instead of treating them as separate processes. The key idea is to execute tests until a failure is observed, then immediately identify failure-inducing interactions, and use this information to guide subsequent test case generation. This framework is useful in a testing environment where efficient identification of failure-inducing interactions is very important. It can be applied to complex systems with numerous interactions and configurations, hence making it suitable for both industrial and academic applications.

Blue, et al., [38] talks about an end-to-end automated solution for Combinatorial Test Design (CTD) that is based on test optimization, TCs generation, TCs execution and FL. This solution is implemented on an industrial framework. The novelty of this solution is the Inverse CTD algorithm. This algorithm assumes that FICs have the maximum length of the strength of the test plan. It also assumes that every two FICs share at least one parameter. The Inverse CTD algorithm detects FICs efficiently by utilizing the executed test plan results and excluding values appearing in failing TCs.

Qi, et al., [39] proposes a new algebraic system called Test Algebra (TA). This is designed to help identify faults in CT for software-as-a-service (SaaS) applications. The TA framework uses algebraic rules to combine test results from different servers in a cloud environment. This helps in parallel and distributed testing.

Ghandehari, et al., [2] [7] presents an FL approach leveraging FICs identified through CT. The paper introduces BEN (Bug Exposing Node), a CT based FL tool which leverages CT results to rank statements based on their likelihood of being faulty. It is particularly useful for complex software systems with numerous configurations and interactions, providing a systematic approach to fault localization.

Some more recent work on CT and FL can be found in

2.1 Methodology

From the literature survey and domain knowledge, specific heuristics for identifying the FICs in combinatorial testing have been identified. They are as follows:

1) For every combinatorial test case with a strength “t” that fails, there exists a test case with strength t = 2 (pair-wise testcase) that will also fail.

2) If a failed test case exhibits an interaction between parameter values that is also found in a passed test case, then this specific combination does not introduce failures, hence it can be termed as *not failure-inducing*.

3) If the pair-wise parameter value combination is not found in any passed test case, it is more likely to be *failure-inducing*.

From the above identified assumption, certain rules are derived to help identify the FICs. They are:

Rule for CT:

if (a combinatorial test case of test suite strength *t* is FAIL)

then (there exists a parameter combination of strength *t = 2* in the failed TC);

Rule for FL in CT:

if (failed_TC_parameter_combination NOT_EQUAL_TO passed_TC_parameter_combination)

then (failed_TC_parameter_combination is LIKELY_FAILURE_INDUCING)

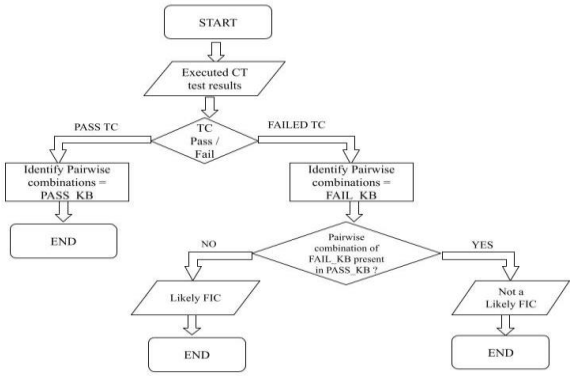
else

check the next parameter combination

2.2 Rule-Based Approach

Using the above-mentioned rules, an approach was developed for identifying the Likely FICs which is given in Fig 1. The approach assumes that the FICs are always pair-wise combinations, in other words the combinations that are likely to cause a failure are always 2 parameter combinations.

Fig 1: Rule-based approach for identifying the Likely FICs.



The proposed approach was implemented using Python and tested by giving different inputs. The complexity of the approach has been calculated and analysed in a way that is computationally viable for practical applications. The developed approach was further validated and is presented as case studies which is presented in section 3.

3. CASE STUDIES

The case study of the different inputs given and the analysis of the results obtained is as given.

3.1 Case Study 1: Validating the rule-based approach on a sample input

The first case study is taken from Nishiura Kinari, et.al, [24], where the parameter combinations identified are used to build a test suite which is then submitted as input to the two approaches proposed in the paper, namely- FROGa and FROGb. These algorithms were developed for localization of faulty interactions using logistic regression analysis. Since the proposed approach in the current paper is similar, we have used the same input here.

The approaches proposed in [24] used a sample SUT with parameters and their values as CPU type, Network type, DBMS, Operating system, Browser. Table 1 depicts parameters and their values.

Table 1. Parameters and their values given as input for combinatorial testing [24]

Parameter	Description	Values
p1	CPU	Intel (=1), AMD (=2)
p2	Network	Wifi (=1), LAN (=2)
p3	DBMS	MySQL (=1), Sybase (=2)
p4	OS	Win (=1), Linux (=2), Mac (=3)
p5	Browser	IE (=1), Firefox (=2), Chrome (=3)

Certain constraints were applied for the valid test cases such as:

1. If the OS is Mac, then the CPU cannot be AMD.

2. If the Browser is IE, then the OS must be Win.

3.1.1 Test Input with execution results

The input from table 1 was used to build a Covering array that realises a 3 way test of the SUT - along with the results. The combinatorial test cases thus generated along with their results are shown in table 2. This input was submitted to the Rule-based approach developed.

Table 2. CT test suite execution results of sample input [24]

Test #	p1	p2	p3	p4	p5	Status
1	2	2	1	2	3	Pass
2	1	2	2	1	1	Pass
3	2	1	1	1	2	Pass
4	1	2	2	3	2	Pass

5	2	1	2	1	1	Pass
6	1	1	2	3	3	Pass
7	2	2	2	1	2	Pass
8	1	2	1	1	1	Fail
9	1	1	1	3	2	Pass
10	1	2	1	3	3	Pass
11	1	1	2	2	2	Fail
12	1	2	1	2	3	Fail
13	2	2	2	1	3	Pass
14	1	1	1	1	3	Pass
15	2	1	2	2	3	Pass
16	1	1	1	1	1	Pass
17	1	2	1	1	2	Pass
18	2	1	1	2	2	Pass
19	2	2	2	2	2	Pass
20	2	2	1	1	1	Fail

3.1.2 Application of rules for identifying Likely FICs

The approach developed was applied on the input given in table 2. The pairwise (t=2) parameter-value combinations present in the FAIL_KB but not in the PASS_KB were identified as:

((Parameter1, value_Parameter1), (Parameter2, value_Parameter2))

((p1, 1), (p4, 2))

((p2, 2), (p5, 1))

These combinations didn't show up in the PASS test cases. The rules suggested they might be *Likely FICs*. This case study validated how well the rule-based fault localization approach worked. The rule-based approach gave a step-by-step effective way to identify failure inducing combinations. This validation also shows that the approach proposed is effective and could be used for other applications as well.

3.2 Case Study 2: Three factor authentication system

The case study involved a project on three-factor authentication [40]. The project incorporated facial recognition, eye blink detection and OTP generation as the 3 factors for authenticating a user. For this project, combinatorial test cases were generated using the Microsoft PICT tool. Generated test cases are automated with the aid of the pytest framework, which is one of the most commonly used frameworks to write simple and scalable test cases in Python. This way, the input combination test cases on the login and sign-up pages are tested exhaustively to ensure the proper functioning of the system. This was a real-world project which had different parameters interacting with each other to ensure successful authentication or not. This provided a good option to do CT. The results obtained after CT helped us in using them as an input to our proposed approach. The proposed approach successfully identified likely failure-inducing combinations for this case study.

3.2.1 Test input with execution results

Input data for the combinatorial test cases used were face recognition, keyboard loaded, cursor position, PIN entry, PIN verification, OTP sent, and OTP verification. Table 3 demonstrates the execution of various test cases to verify the functionality and robustness of the three-factor authentication system.

Table 3. CT test suite for 3FA

TC ID	Face recognition	Keyboard loaded	Cursor	PIN entry	PIN verification	OTP sent	OTP verification	Result
1	TRUE	TRUE	TRUE	TRUE	Correct	TRUE	Correct	Pass
2	TRUE	TRUE	TRUE	FALSE	-	-	-	Pass
3	TRUE	TRUE	FALSE	TRUE	Incorrect	FALSE	-	Fail
4	TRUE	FALSE	FALSE	FALSE	-	-	-	Pass
5	FALSE	TRUE	FALSE	FALSE	-	-	-	Fail
6	TRUE	TRUE	TRUE	TRUE	Incorrect	FALSE	-	Pass
7	FALSE	TRUE	FALSE	FALSE	-	-	-	Pass
8	TRUE	TRUE	TRUE	TRUE	Correct	FALSE	-	Fail
9	TRUE	TRUE	TRUE	TRUE	Correct	TRUE	Incorrect	Pass
10	TRUE	TRUE	FALSE	TRUE	Correct	TRUE	Correct	Pass

3.2.2 Application of rules for identifying Likely FICs

For the input identified in table 3, the rule-based approach was applied to identify the Likely FICs. The pairwise combinations that are likely FICs thus identified are:

((Parameter1, value_Parameter1), (Parameter2, value_Parameter2))

((Cursor, FALSE), (PIN verification, Incorrect))

((Cursor, FALSE), (OTP sent, FALSE))

((PIN Verification, OTP sent), (Correct_PIN verification, FALSE))

These combinations were found to be likely causing the failures, helping to narrow down the debugging efforts effectively. The FICs judged as such according to the rules did not appear in the passing TCs. In this paper, the rule-based approach for identifying the likely FICs has been successfully applied to a three-factor authentication system. The combinatorial TCs, which are generated by Microsoft PICT and automated with pytest, are effective at finding FICs. The rule-based approach permitted FL in a systematic and effective way, making the authentication system more robust and reliable. This case study provides proof of the effectiveness of the developed approach in practice and points toward its potential for more general use within CT.

4. DISCUSSION

The efficacy of the rule-based approach for identifying the Likely FICs has been demonstrated in this application to the three-factor authentication project and the validation using the sample input provided in the Software Quality Journal paper. The FICs that were found were correct, which validates the correctness and reliability of the approach for future combinations of factors. Besides, it is the use of rules specific to localizing FICs that greatly increased the effectiveness of identifying problematic interactions.

The potential of the approach has proven to be helpful in a way that it can identify likely FICs without exhaustive testing, thus reducing the time and resources needed for debugging. Particularly, this effectiveness is valuable for complicated systems with large interactions where other methods for identifying FICs are less effective.

The rule-based approach controlled complex interactions between many parameters while providing a full scope of potential failure cases. Being the systematic application of a set of rules to allow identification of rule combinations leading to failures in a way that is hardly accomplished in any other way, it makes this mechanism of utmost importance for robustness and reliability maintenance of software systems, especially of a critical nature like authentication and security.

The use of real-world examples to validate the approach - say, for example, a three-factor authentication project and the case study from the Software Quality Journal paper - established its practical relevance. The correct identification of combinations causing failures in so diverse domains illustrates well the

versatility and robustness of the approach. This, too, gives confidence in coping with other kinds of software systems and testing environments.

The approach has the potential to be much more widely used in various areas of software testing and quality assurance. This way, the technique will easily be useful in industries where software reliability is a key concern to quickly capture FICs. Furthermore, the adaptability of the approach to diverse testing frameworks and environments makes it an invaluable tool toward harnessing software quality across other applications.

5. CONCLUSION

The proposed approach of this research provides a way of using CT and FL rules for the identification of Likely and Not-likely FICs. The developed approach was validated on case studies where it showed its effectiveness in identifying likely FICs and confirmed its practical applicability and reliability. It vastly improved the efficiency in identifying problematic interactions with applied predefined rules, greatly reduced the time and resources for debugging, and actively controlled complex interactions of many input parameters.

Validation performed successfully in real-world scenarios, underlining the versatility and robustness of the approach, giving us confidence that it is able to treat diverse software systems and testing environments. The rules-based approach gives a dependable and practical way to identify likely FICs that are rich in applicability to many fields of software testing and quality assurance. This methodology forms the basis for further research and development within the area of FL in CT, which promises to bring effective advancements in modern software testing and quality assurance practice.

References

[1] C. Nie and H. Leung, “A survey of combinatorial testing,” ACM Computing Surveys (CSUR), vol. 43, no. 2, pp. 1–29, 2011.

[2] L. S. Ghandehari et al., “Identifying failure-inducing combinations in a combinatorial test set,” in 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012, pp. 370–379.

[3] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, “Combinatorial testing approaches: a systematic review,” IRAQI JOURNAL OF COMPUTERS, COMMUNICATIONS, CONTROL AND SYSTEMS ENGINEERING, vol. 22, no. 4, pp. 60–79, 2022.

[4] A. Zeller and R. Hildebrandt, “Simplifying and isolating failure-inducing input,” IEEE Transactions on Software Engineering, vol. 28, no. 2, pp. 183–200, 2002.

[5] L. S. Ghandehari et al., “Fault localization based on failure-inducing combinations,” in 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), 2013, pp. 168–177.

[6] K. Sarjapur et al., “Big data management system for personal privacy using SW and SDF,” in Information Systems Design and Intelligent Applications: Proceedings of Third International Conference INDIA 2016, Volume 2. Springer India, 2016, pp. 757–763.

[7] L. S. Ghandehari et al., “BEN: A combinatorial testing-based fault localization tool,” in 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2015, pp. 1–4.

[8] J. A. Jones, M. J. Harrold, and J. Stasko, “Visualization of test information to assist fault localization,” in Proceedings of the 24th International Conference on Software Engineering, 2002, pp. 467–477.

[9] Z. Zhang and J. Zhang, “Characterizing failure-causing parameter interactions by adaptive testing,” in Proceedings of the 2011 International Symposium on Software Testing and Analysis, 2011, pp. 331–342.

[10] D. R. Kuhn and M. J. Reilly, “An investigation of the applicability of design of experiments to software testing,” in 27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002, pp. 91–95.

[11] M. Renieres and S. P. Reiss, “Fault localization with nearest neighbor queries,” in 18th IEEE International Conference on Automated Software Engineering, 2003, pp. 30–39.

[12] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, “Software fault interactions and implications for software testing,” IEEE Transactions on Software Engineering, vol. 30, no. 6, pp. 418–421, 2004.

[13] D. Bisht, Dipti, G. Kotecha, M. Jha, and R. Jayaram, “Predicting a faulty parameter using machine learning - an analysis,” International Journal of Creative Research Thoughts (IJCRT), vol. 11, no. 3, pp. 2320–2882, June 2023.

[14] A. Zeller, “Isolating cause-effect chains from computer programs,” ACM SIGSOFT Software Engineering Notes, vol. 27, no. 6, pp. 1–10, 2002.

[15] G. Chaithanya *et al.*, “Analysis of Combinatorial Testing,” *International Journal of Creative Research Thoughts*, vol. 9, no. 6, Jun. 2021. [Online]. Available: issn: 2320-2882.

[16] R. Jayaram and R. Krishnan, “Approaches to fault localization in combinatorial testing: A survey,” in Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 2. Springer Singapore, 2018, pp. 491–502.

[17] R. Jayaram and R. Krishnan, “Combinatorial Test Perspective on Test Design of SDN Controllers,” in *Proceedings of Second International Conference on Sustainable Expert Systems (ICSES 2021)*, Springer Nature Singapore, 2022, pp. 57–70.

[18] Nie, Changhai, and Hareton Leung. "The minimal failure-causing schema of combinatorial testing." ACM Transactions on Software Engineering and Methodology (TOSEM) 20.4 (2011): 1-38.

[19] Li, Jie, Changhai Nie, and Yu Lei. "Improved delta debugging based on combinatorial testing." 2012 12th International Conference on Quality Software. IEEE, 2012.

[20] Niu, Xintao, et al. "An interleaving approach to combinatorial testing and failure-inducing interaction identification." IEEE Transactions on Software Engineering 46.6 (2018): 584-615.

[21] Fouché, Sandro, Myra B. Cohen, and Adam Porter. "Incremental covering array failure characterization in large configuration spaces." Proceedings of the eighteenth international symposium on Software testing and analysis. 2009.

[22] Shakya, Kiran, et al. "Isolating failure-inducing combinations in combinatorial testing using test augmentation and classification." 2012 IEEE fifth international conference on software testing, verification and validation. IEEE, 2012.

[23] X. Niu et al., “An interleaving approach to combinatorial testing and failure-inducing interaction identification,” IEEE Transactions on Software Engineering, vol. 46, no. 6, pp. 584–615, 2018.

[24] K. Nishiura, E. H. Choi, E. Choi, and O. Mizuno, “Two improving approaches for faulty interaction localization using logistic regression analysis,” Software Quality Journal, vol. 32, no. 1, pp. 1–35, 2024.

[25] Wang, Ziyuan, et al. "Adaptive interaction fault location based on combinatorial testing." 2010 10th International Conference on Quality Software. IEEE, 2010.

[26] Zheng, Wei, et al. "Locating minimal fault interaction in combinatorial testing." Advances in Software Engineering 2016.1 (2016): 2409521.

[27] P. Arcaini, A. Gargantini, and M. Radavelli, “Efficient and guaranteed detection of t-way failure-inducing combinations,” in *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2019, pp. 1–10.

[28] M. G. Chakravarthy *et al.*, “Identification of Failure Inducing Combinations using Greedy Heuristic Algorithm,” in *International Conference on Advanced Data-driven Intelligence and Engineering (ICADIE-2024)*. Springer, 2024 (in press).

[29] Zhang, Jian, Feifei Ma, and Zhiqiang Zhang. "Faulty interaction identification via constraint solving and optimization." International Conference on Theory and Applications of Satisfiability Testing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[30] Martínez, Conrado, et al. "Locating errors using ELAs, covering arrays, and adaptive testing algorithms." SIAM Journal on Discrete Mathematics 23.4 (2010): 1776-1799.

[31] Hagar, Jon D., et al. "Introducing combinatorial testing in a large organization." Computer 48.4 (2015): 64-72.

[32] Nagamoto, Takahiro, et al. "Locating a faulty interaction in pair-wise testing." 2014 IEEE 20th Pacific Rim International Symposium on Dependable Computing. IEEE, 2014.

[33] Jin, Hao, et al. "A satisfiability-based approach to generation of constrained locating arrays." 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2018.

[34] Ryan Lekivetz and Joseph Morgan. “Covering arrays: Using prior information for construction, evaluation and to facilitate fault localization”. In: Journal of Statistical Theory and Practice 14.1 (2020), pp. 1–20.

[35] H. Jin, C. Shi, and T. Tsuchiya, “A weight-based combinatorial test case prioritization,” Information and Software Technology, vol. 100, pp. 157–169, 2018.

[36] J. Bonn, K. Fögen, and H. Lichter, “A framework for automated combinatorial test generation, execution, and fault characterization,” in 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2019, pp. 224–233.

[37] L. Kampel, D. E. Simos, D. R. Kuhn, and R. N. Kacker, “An exploration of combinatorial testing-based approaches to fault localization for explainable AI,” Annals of Mathematics and Artificial Intelligence, vol. 94, no. 1, pp. 1–14, 2022.

[38] D. Blue, A. Hicks, R. Rawlins, and R. Tzoref-Brill, “Practical fault localization with combinatorial test design,” in 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2019, pp. 268–271.

[39] G. Qi, W. T. Tsai, C. J. Colbourn, J. Luo, and Z. Zhu, “Test-algebra-based fault location analysis for the concurrent combinatorial testing,” IEEE Transactions on Reliability, vol. 67, no. 3, pp. 802–831, 2018.

[40] Rekha Jayaram, Ramya M, Sandhya A, Sinchana M K, Tejashree R, "User Authentication Through Eye Blink-Based PINS Using Histogram Of Oriented Gradients Algorithm", in International Conference on Innovation & Novelty in Engineering and Technology (INNOVA-2024), 20-21, DECEMBER 2024 (in press).