

An efficient test suit reduction methodology for regression testing

Shailendra Gupta, Jitendra Choudhary

Department of Computer Science, Medi-Caps University Indore, Indore, India

Article Info

Article history:

Received Nov 20, 2023

Revised Feb 19, 2024

Accepted Feb 20, 2024

Keywords:

Method of experiment

Minimization methods

Test case

Test case generation

Test suite

ABSTRACT

This paper goal is to provide a more effective algorithm for reducing the amount of test cases in a test suit during the regression testing phase. This algorithm divides the entire test suit into equivalence classes at first then apply boundary value coverage to select test case out of repeated test cases with same importance in suit. This algorithm is based on the concept that before selecting best test cases out of repeated test case in test suit to prepare reduced test suit we can divide all test cases in number of equivalence classes to reduce test cases by great extent. This paper proposed a method of experimentation involving test cases from different software application areas. Furthermore, we discuss a case study to verify and check new algorithm for its efficiency, for that we apply our algorithm on one of the program or group of programs. This complete proposed methodology shall be applied to different software applications belonging to soft computing, engineering software, and financial software. This wok has not been done earlier as found in the literature survey performed by us. Minimization in test cases would lead to lesser testing effort and desirable test completion.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Shailendra Gupta

Department of Computer Application, Medi-Caps University Indore

Madhya Pradesh, 452012, India

Email: Shailendra.gupta@medicaps.ac.in

1. INTRODUCTION

It is well known fact that regression testing is main part of testing phase in software development cycle. It is very obvious that when we modify the code or do any kind of addition in existing code, we require more and more test cases to detect error due to these new addition and changes to the code. As the test suit size increases this process consume lot of time to test effect of changes on existing functionality along with new functionality. Therefore, it is imperative to speed up software development in order to lower how many tests are currently included in the test suite, allowing it to cover all necessary functionality with a lesser number of test instances. Test suit reduction is the name of this procedure [1], [2].

When we begin identifying a subset of currently available test cases to satisfy all stages of software testing requirements, it reveals that it is nondeterministic polynomial (NP)-problem [3], [4]. Few algorithms are designed to reduce existing test suit into reduced test suitin [1], [5]–[7]. As there are number of algorithms for this purpose but out of these there are four more important algorithms that can be used to minimize test cases in test suit. The greedy algorithm G [5] used to select test cased in repeating manner so that test scenarios in new minimized test suit become able to test new a group of specifications. The next test scenario will be chosen by the greedy algorithm hunger games search (HGS) [1] if the current test case meets the majority of the conditions. Chen and Lau [7] introduced an algorithm GRE with three main components i) select all most important test cases, ii) remove all 1-to-1 repeated test cases, and iii) select from remaining

test cases which are need to satisfy remaining requirement. Liu [8] replaced a random choice method for identically covered test scenarios capacity with an existing step to choose identical-cardinality test scenarios in accordance with the covering of boundaries methodology.

As we know that existing algorithms minimize the test cases in test suit but a side effect this test suit is facing problem of fault detection capability loss [9]. Due to this, Jeffrey and Gupta [9], in their paper told that add some repeated test case to reduced test suit to gain same fault detection capability. But according to Lin and Huang [10], it is purely increased maintaining overhead, so they suggested using another testing criteria to select test from test case with equal importance instead of random selection. It is not easy to select new testing criteria to support the process; it is because new criteria may be effective with in some case and has no effect on other programs.

Chvatal [5] and then Leiserson *et al.* [6] have proposed an algorithm for minimum set cover problem. The greedy test suite reduction strategy: Choose a test case that covers a lot of criteria, take out all the requirements it addresses, and until every condition is met, repeat the process. Finding necessary instances of testing that can fulfill certain test criteria that cannot be satisfied in additional test situations is the next approach recommended by Harrold *et al.* [1] for minimizing the test suite. It is algorithm to search each and every test scenario that is covering those test requirement not covered by other test cases. This algorithm repeats its steps until all the requirements are fulfill. By using this method, you can remove the test scenarios between 19% and 60% when compared to the initial number of test instances.

Chen and Lau [7] provide a fresh GRE trick that is broken down into three parts: first is the greedy approach, the required method, and the 1-to-1 repetition technique. The greedy approach is also choosing the test scenarios which fulfill the most of test specifications that have not yet been satisfied, just like the basic greedy approach [5], [6]. The required technique picks all necessary test scenarios, and using the 1:1 approach, duplicate test cases in all are removed. In a simulation, GRE consistently outperforms H in both our studies and the tests conducted [11], [12]. However, GRE and H both have the capacity to reduce test suites.

A novel test suite reduction method was proposed by Jeffrey and Gupta [9] hence, by preserving some duplicate test cases depending on additional testing standards, can improve the capacity to detect faults. When a test case was found as duplicate with regard to testing condition C, its duplicity with regard to the other testing criteria was also examined. Otherwise, it will be kept. Compared to the test suite achieved by meeting a single testing criterion, the reduced test suite is more able to detect errors because it satisfies more testing criteria.

A innovative approach was presented by Sheikh *et al.* [13]. A combination of evolutionary algorithms is provided with Test Reduce in order to find an optimal and compact set of test cases. This study's ultimate objective is to provide a technique that could reduce the number of regression test cases when relevant criteria are available. The selected set will help in assessing the quality of the code and researching its ripple effects.

Jung *et al.* [14] provided a workable process for implementing the software product line (SPL) testing strategy as well as a formula that enhances our earlier technique to reduce the superfluous tracing of test executions and recurrence of test executions. SPL testing may be completed more rapidly with this method. Package level security metrics were used by Tanejaa *et al.* [15] to demonstrate how to assess the probability of errors in software modules that utilize objects. It improves FEP and cuts down on execution time.

The test suite reduction model developed by Zhang *et al.* [16] is transformed into a typical optimization issue. Furthermore, they use the quantum bits approach to encode the chromosome as discrete information. When utilizing the condensed test suite, it can significantly save testing expenses and increase test efficiency. An approach that reduces the number of test cases required by identifying a representative group that satisfies the testing requirements was proposed by Mohapatra and Pradhan [17]. How many test cases with different chromosomal lengths are included in each iteration of a test suit? Regression testing may save costs and increase software productivity with an efficient test suite.

Lawanna [18] developed a unique model to improve the reduction of regression tests, which the research suggests using as an alternative to conventional approaches. There are four algorithms: testing, ordering, reducing test cases, and defect-fixing. It suggests that the size of the fixed bugs is less, at about 40% and 200%, respectively. Section 2 discuss about existing methods along with proposed methodology. Section 3 covering software application areas for proposed methodology. Finally, the concluding remarks will be presented in section 4.

2. EXISTING MINIMIZATION METHODS AND PROPOSED METHODOLOGY DETAILS OF TEST REDUCTION

2.1. Math and algorithm for minimization methods

In this section we describe maths and algorithm used in recent time to perform operation of minimization. In first subsection we discussed about new technique called TestReduce to minimize test cases

in test suit. In second subsection we discussed about an investigation on reducing duplicate SPL testing test executions.

2.1.1. An optimized test case minimization method for regression testing using GA (2023) [13]

TestReduce: to reduce regression TCs, TestReduce is built on genetic algorithm (GA). The goal of test reduce is to find the best possible approach to identify to begin the regression testing process, a specific collection of test scenarios must be selected. The chosen set will assist in evaluating the modified code's quality and in conducting a ripple impact analysis. This section lists the several phases that make up the test reduce.

Step 1: requirement prioritization.

Step 2: requirements associations.

Step 3: correction of the module bugs.

Step 4: association level of rectified modules.

Step 5: derived objective function: the importance of the objective function is crucial because test case priority is based on it. In this study, the goal function was a combinatorial problem, as demonstrated in (1).

$$f p, da, di, dr, dm = (p + da + di + dr + dm) - 5 \quad (1)$$

Step 6: application of genetic algorithm.

2.1.2. An investigation on reducing duplicate SPL testing test executions (2022) [14]

A minimum set of steps are included in the suggested application procedure in order to apply our strategy to SPL testing. Therefore, extra stages for test case minimization and prioritizing, change impact analysis, and are applied. The method can be expanded to include everything required for thorough testing of a product line.

Step 1: get the product family.

Step 2: assemble/update the checksum matrix should not be used in the text. Instead, additional information can be added to the reference list.

Step 3: decide which test scenario to run.

Step 4: carry out test scenario and get execution traces is depicted in Figure 1.

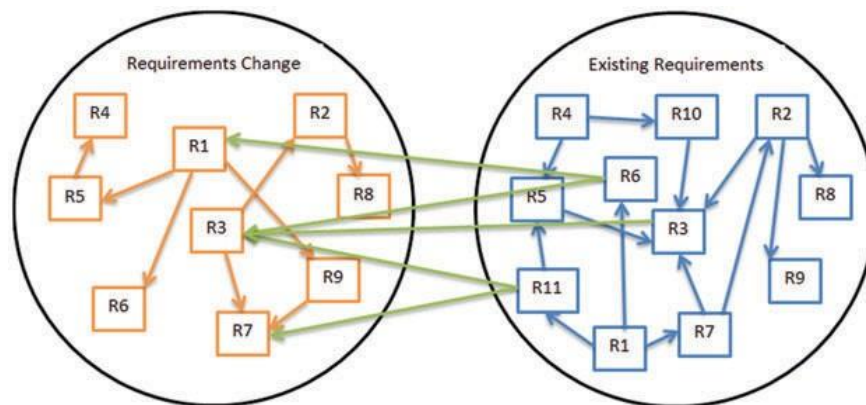


Figure 1. SPL

So, in proposed algorithm, we apply equivalence class partition before applying boundary value condition, it will reduce the test suit with great extent. So, we present a novel algorithm for test case reduction by applying equivalence class partition as first step for test suit reduction. New algorithm extends the algorithm given by Liu by introducing equivalence partition in algorithm. We also present the discussion of the implementation of new algorithm with case study to check the algorithm.

2.2. Proposed method details of test reduction

The proposed methodology is depicted in Figure 2 as a flowchart. The flowchart consists of five main phases as: program collection to apply methodology, procedures to find set of test cases, list of test case

minimization procedures, and area of study on minimized test cases. The last is the application of minimized test cases on different software's.

2.2.1. In methodology shown in Figure 2 we follow these steps

Initially we select N number of program containing conditions.

Simple conditions examples usually found in codes:

- If (a<b)
- If (a>b)
- If (a<=b)
- If (a>=b)
- If (a==b)
- If (a!=b)

Compound conditions examples usually found in codes:

- If ((a<b) && (c>d))
- If ((a<b) || (c>d))

AND is represented with &&

OR is represented with ||.

2.2.2. After program selection we will generate different test cases using

Boundary value analysis: analysis-this method checks for problems at the input level and values can range from minimum to maximum, just inside, and outside. It is an additional method for designing test cases to equivalence class partitioning. In contrast to other approaches, this one draws the test cases from both the input conditions and the output domain. Equivalence class partition: with this technique, by categorizing or dividing the input domain of the software unit, test cases are produced. As a result, fewer test instances are run overall to a manageable quantity as a result. An analogous class defines a collection of valid and invalid states for the input condition.

2.2.3. Techniques for reducing test cases [19]

- A. Requirement based: using the fewest number of test cases to satisfy all testing criteria is the fundamental goal of test suite reduction. One such method is to use requirement optimization to build test cases based on the requirements.
- B. Coverage based: the primary goal of the coverage-based reduction technique is to guarantee that the most paths possible inside a particular program are carried out. Case base reasoning (CBR) is used to accomplish this. CBR is divided into three categories: pivotal, auxiliary, and case.
 - Problems are solved using case-based searches for the most comparable problems in a memory.
 - Although the deletion of an auxiliary-based instance doesn't alter competency, it does have an impact on the If eliminated, a pivotal-based case directly affects the system's capability.
- C. GA: it is a method of solving test case reduction issues like evolutionary computation that is based on computational intelligence.

The subsequent actions were taken:

- Test case runtime and coverage were used to derive fitness value.
- The smaller suite could contain just tests that made sense.
- Up till a test suite is found to be optimized, this process is repeated.
- The findings demonstrated the generality and effectiveness of the recommended test suite reduction technique.

This strategy offers various advantages, one of which is that it reduces the several test cases while also shortening the overall runtime. However, it is ineffective when the defect detecting capability and other CR.

- D. Clustering: fewer test scenarios in the test suite, for efficiency and increase performance, the data mining approach of accumulating techniques is utilized. Instead of using all of the test cases generated by independent routes, clustering makes it possible to check the program using just one of the clustered test cases.
- E. Greedy algorithm: the popular code-based reduction technique is called t, it is employed while creating test suites using model-based techniques. The test scenarios are selected based on their ability to satisfy the most unmet requirements. Repeat this process until the entire test suite's test scenarios result in the creation of a smaller test suite. The connection between test cases and testing specifications is the foundation for how this algorithm works.

The advantage of the greedy algorithm is that it greatly lowers the total number of test instances, but in the event of a tie, a random selection of test instances is made.

- F. Fuzzy logic: the use of fuzzy logic is a different method for test suite optimization. Given that it speeds up execution time and reduces the size of the regression testing, this method is regarded as safe. Testing is carried out in this case to a is founded on an objective process that is remarkably similar to human judgment. Many times, test suites are optimized and examined for safe reduction using CI-based techniques, which may then be completed using control flow diagrams. For traversing tests for the best solutions, these graphs are employed. Since it is believed to be more secure than alternative methods for performing regression testing, this methodology is widely recommended.
- G. Program slicing: it is a method for building a slice set and checking a programmed against a particular property.
 Three different slicing methods exist:
 - Static slicing
 - Dynamic slicing
 - Relevant slicing
- H. Hybrid algorithm: this algorithm uses the efficient procedure such as genetic algorithm approximation and the greedy strategy to build superior Pareto fronts that can be used to achieve a variety of objectives. The test criterion is here thought of as being mathematically described by the objective functions. The greedy method is modified to be more economical for statement coverage and computational cost. Execution time, code coverage, and fault coverage are also taken into account for fault detection optimization.

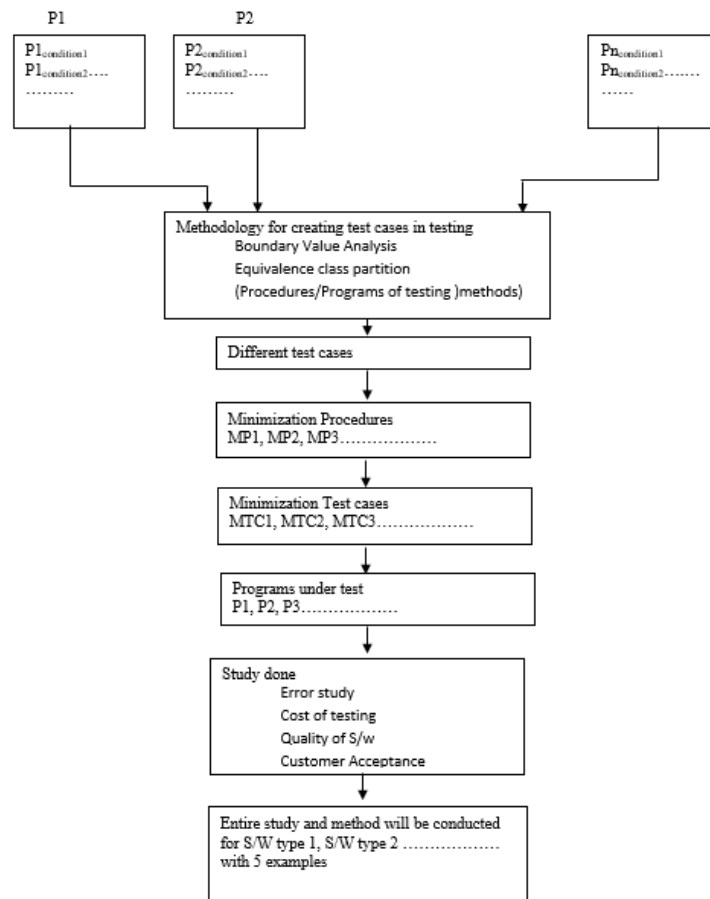


Figure 2. Method of experiment

- I. Apply minimized test suit on list of N programs selected initially to study different parameters like error study, cost of testing, and quality of s/w and customer acceptance.
- J. Above methodology is applied to different s/w types with at least 5 examples shown in Figure 3.

In Figure 3 $P_1^1, P_2^1, P_3^1, P_4^1, P_5^1$ are five different programs to be applied upon by the methodology one at a time. They all belong to software type 1. Above statement is equally applicable for all other programs of each software type. So, in all the methodology will be applied 5×5 times on 25 different programs of 5 software types.

If methodology has:

- Number of test case generation methods (TCGM)
- Number of minimization methods (MM)

One TCGM and one MM will work at a time. So, in all these shall be $5 \times 5 \times a \times b$ runs of the methodology and each run shall generate 5 different results as mentioned.

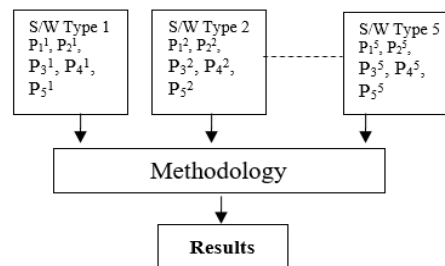


Figure 3. Application of methodology

3. APPLICATION AREAS OF METHOD

3.1. Soft computing applications

Computing techniques have undergone a significant paradigm change that parallels the astounding expansion of the human mind. The shift from hard computing to soft computing is one-key advancement we have observed. Most of the techniques used in soft computing were derived from various mathematical concepts, like fuzzy logic. Unlike classical computing, which emphasizes objective truths and approximations, soft computing, is the technique of resolving complicated real-world issues using approximations rather than precise computations.

Soft computing component parts are:

- FL, or fuzzy logic.
- Computer evolutionary (EC)
- Automated learning (ML).
- Probability-related reasoning (PR) [20].

List of soft computing applications:

- Soft computing for recognition of handwritten script.
- Data compression, picture processing, and soft computing.
- Soft computing in automotive systems and manufacturing.
- Soft computing techniques in bioinformatics.

3.2. Software engineering software

There are several responsibilities in the large field of software engineering, depending on how complicated the program is. As a result, the teams of engineering functions contain a variety of software engineers. The roles that software engineers most frequently play include the following [21]: i) UI/UX engineer, ii) engineer in the backend, iii) engineering full stack, iv) software quality assurance engineer, v) in-test software development engineer, and vi) engineer with DevOps.

3.3. Financial software

Since 2009, MindK has worked with many start-up companies which offer financial services, such as: i) online payments, ii) invoices, iii) insurance, iii) investment risk analysis, and iv) pension calculation. We selected the most promising financial software categories, looked at their features, and based on our knowledge and thorough market research. Examples of the successful firms that used them [22]: i) accounting programs, ii) transaction gateways, iii) retirement and insurance, iv) wallets and mobile transactions, and v) financing apps.

3.4. Cloud applications

By utilizing cloud computing, information technology (IT) teams and developers can concentrate on what is important rather than performing mundane tasks like: i) capacity, ii) planning, iii) maintenance, and

iv) buying. As cloud computing has gained popularity and become more widely used, a variety of concepts and deployment strategies have arisen to fulfill a range of user requirements. You have different levels of management, flexibility, and control based on the deployment strategy and the cloud service you pick. Understanding the variations between platform infrastructure as a service, software as a service, and you will be able you select the set of services best suited to your requirements with the aid of deployment plans and other information [23].

3.5. Business applications

A number of firms employ various types of business application software to automate their procedures for increasing operational effectiveness and gather important data about front- and back-office activities. Research by Adobe found that 25% of business owners who used automated technologies prioritized big data analysis while 30% put time savings first. Employees can focus their energies on more difficult jobs by saving time on routine chores, which improves workflow. Organizations are able to concentrate on boosting their earnings and expanding thanks to the increased efficiency and access to data reports [24].

3.6. Artificial intelligence and machine learning

Any specific tool used for ML classifiers, unsupervised learning, self-iteration based on data, and artificial intelligence (AI) is referred to as machine learning (ML) software. Many pieces of software used in business today, include ML in applications like email filtering and computer vision. There is additional software for simulation, hiring, architecture, and accounting that is specifically made for ML. Certain ML toolkits, including those in this article's list, can be specially designed to fit your particular data sets and process requirements [25].

3.7. Data analytics

Data can be used to find patterns and draw conclusions about the information they hold, data sets are analyzed using data analytics (DA). Increasingly frequently, the utilization of specialist hardware and software is required for data analytics. In the commercial sector, technology and methods for data analytics are frequently utilized to assist organizations in making more informed business judgments. Researchers and scientists also employ analytics tools to confirm or disprove scientific models, theories, and hypotheses.

4. CONCLUSION

Test case minimization has been a focus area look at the enormous time and cost involved to achieve full test case completion criteria. Our work proposes test case minimization for regression testing. Our work proposes methodology to test soft computing, financial and other application which are emerging and having industry values. They all if subjected to minimization procedures shall be tested in time; in designated cost and quality required. This methodology is proposed by us has been designed keeping in mind the various reveling and promising software engineering tools; financial s/w, cloud application; business applications; AI/ML applications and data analytical s/w. No such earlier work has been found in literature. So as to perform any type of comparison at this stage depicting the uniqueness and originality of our proposed experimental method.





REFERENCES

- [1] M. J. Harrold, R. Gupta, and M. Lou Soffa, "A methodology for controlling the size of a test suite," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 2, no. 3, pp. 270–285, Jul. 1993, doi: 10.1145/152388.152391.
- [2] H. Miao, P. Liu, J. Mei, and H. Zeng, "A new approach to automated redundancy reduction for test sequences," in *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2009*, Nov. 2009, pp. 93–98, doi: 10.1109/PRDC.2009.23.
- [3] M. R. Garey and D. S. Johnson, *Computers and Intractability: a guide to the theory of NP-completeness*. WH Freeman & Co., 1979.
- [4] T. Y. Chen and M. F. Lau, "Dividing strategies for the optimization of a test suite," *Information Processing Letters*, vol. 60, no. 3, pp. 135–141, Nov. 1996, doi: 10.1016/S0020-0190(96)00135-4.
- [5] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, Aug. 1979, doi: 10.1287/moor.4.3.233.
- [6] C. E. Leiserson, R. L. Rivest, C. Stein, and T. H. Cormen, "Introduction to algorithms," *The MIT press*, 2011.
- [7] T. Y. Chen and M. F. Lau, "A new heuristic for test suite reduction," *Information and Software Technology*, vol. 40, no. 5–6, pp. 347–354, Jul. 1998, doi: 10.1016/s0950-5849(98)00050-0.
- [8] P. Liu, "An efficient reduction approach to test suite," in *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, Jun. 2014, pp. 1–5, doi: 10.1109/SNPD.2014.6888743.





- [9] D. Jeffrey and N. Gupta, "Improving fault detection capability by selectively retaining test cases during test suite reduction," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 108–123, Feb. 2007, doi: 10.1109/TSE.2007.18.
- [10] J. W. Lin and C. Y. Huang, "Analysis of test suite reduction with enhanced tie-breaking techniques," *Information and Software Technology*, vol. 51, no. 4, pp. 679–690, Apr. 2009, doi: 10.1016/j.infsof.2008.11.004.
- [11] Z. Hao, Z. Lu, and M. Hong, "An experimental comparison of four test suite reduction techniques," in *Proceedings - International Conference on Software Engineering*, May 2006, vol. 2006, pp. 636–639, doi: 10.1145/1134285.1134380.
- [12] H. Zhong, L. Zhang, and H. Mei, "An experimental study of four typical test suite reduction techniques," *Information and Software Technology*, vol. 50, no. 6, pp. 534–546, May 2008, doi: 10.1016/j.infsof.2007.06.003.
- [13] R. Sheikh, M. I. Babar, R. Butt, A. Abdelmaboud, and T. A. E. Eisa, "An optimized test case minimization technique using genetic algorithm for regression testing," *Computers, Materials and Continua*, vol. 74, no. 3, pp. 6789–6806, 2023, doi: 10.32604/cmcc.2023.028625.
- [14] P. Jung, S. Kang, and J. Lee, "Reducing redundant test executions in software product line testing-a case study," *Electronics (Switzerland)*, vol. 11, no. 7, p. 1165, Apr. 2022, doi: 10.3390/electronics11071165.
- [15] D. Taneja, R. Singh, A. Singh, and H. Malik, "A Novel technique for test case minimization in object oriented testing," *Procedia Computer Science*, vol. 167, pp. 2221–2228, 2020, doi: 10.1016/j.procs.2020.03.274.
- [16] Y. K. Zhang, J. C. Liu, Y. A. Cui, X. H. Hei, and M. H. Zhang, "An improved quantum genetic algorithm for test suite reduction," in *Proceedings - 2011 IEEE International Conference on Computer Science and Automation Engineering, CSAE 2011*, Jun. 2011, vol. 2, pp. 149–153, doi: 10.1109/CSAE.2011.5952443.
- [17] S. K. Mohapatra and M. Pradhan, "Finding representative test suit for test case reduction in regression testing," in *2015 International Conference on Computer, Communication and Control (IC4)*, Sep. 2015, pp. 1–6, doi: 10.1109/IC4.2015.7375657.
- [18] A. Lawanna, "Extended regression test model for test suite reduction," in *2017 IEEE 6th Global Conference on Consumer Electronics, GCCE 2017*, Oct. 2017, vol. 2017-January, pp. 1–5, doi: 10.1109/GCCE.2017.8229217.
- [19] S. Felice, "Test case reduction and techniques to follow," *browserstack.com*, 2022. [Online] Available: <https://www.browserstack.com/guide/test-case-reduction-and-techniques>.
- [20] J. Hoffman, "Applications of soft computing," *wisdomplexus.com*, Oct. 2019. [Online] Available: <https://wisdomplexus.com/blogs/applications-soft-computing/>.
- [21] S. Singh "Different types of software engineer roles", *browserstack.com*, May 2023. [Online]. Available: <https://www.browserstack.com/guide/what-are-the-different-types-of-software-engineer-roles>.
- [22] "12 types of financial software to build a successful fintech startup in 2024", *mindk.com*, August 2023. [Online]. Available: <https://www.mindk.com/blog/types-of-financial-software/>.
- [23] "Cloud computing applications", *javatpoint.com*, August 2023. [Online]. Available: <https://www.javatpoint.com/cloud-computing-applications>.
- [24] "8 different types of business application software", *anyconnector.com*, July 2020, August 2023. [Online]. Available: <https://anyconnector.com/actionable-insights/different-types-of-business-application-software.html>.
- [25] J. Boog, "16 best machine learning software reviewed for 2024", *theqalead.com*, Ja. 2024. [Online] Available: <https://theqalead.com/tools/best-machine-learning-software/>.

BIOGRAPHIES OF AUTHORS



Mr. Shailendra Gupta     was born in 1976 at Murena, M.P. India. He received his B.Sc. degree in Computer Science from PMB Gujrati Science collage, Indore in 1997, MCA. Degree in Computer Science in 2001 and pursuing his Ph.D. Degree from Medi-Caps University Indore. He received his Ph.D. degree from Devi Ahilya University Indore in 2014. His areas are software engineering and software testing. He has published 3 to 4 research paper in reputed international journal and conferences. He is an Assistant Professor in CA Department at Medi-Caps University, Indore, M.P., India. He has 23 years of teaching work experience at the UG and PG levels. He can be contacted at email: Shailendra.gupta@medicaps.ac.in.



Dr. Jitendra Choudhary     was born in 1984 at Dewas, M.P. India. He received his B.Sc. degree in Computer Science from Holkar Science College, Indore in 2003, M.Sc. degree in Computer Science in 2005 and M.Tech. degree in Computer Science (with Distinction) in 2010 from SCSIT, Devi Ahilya University Indore. He received his Ph.D. degree from Devi Ahilya University Indore in 2014. His areas are software engineering and software testing. His research area includes extreme programming and software maintenance. He has published more than 20 research paper in reputed international journal and conferences. He has received Gold Medal (AIR-1) in Software Engineering course run by IIT Kharagpur through Swayam-NPTEL. He is an Associate Professor and HOD, CS at Medi-Caps University, Indore, M.P., India. He has 17 years of teaching work experience at the UG and PG levels. He can be contacted at email: jitendra.scsit@gmail.com.