ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Combinatorial First Order Polynomial Coverage based Test Suites Prioritization for Improving Software System Quality

M.Bharathi
Department of Computer Science
Periyar University College of Arts and Science,
Pennagaram, Tamilnadu,India

Dr.V.Sangeetha
Department of Computer Science
Periyar University College of Arts and Science
Pappireddipatti, Tamilnadu,India

Abstract: Combinatorial Testing (CT) is performed to ensure the development of a software system quality. In CT, many research works has been designed for test suite minimization. But, existing test suite minimization techniques does not covers the more number of test cases for detecting the maximum faults in software programs. Therefore, there is a requirement for new test suite minimization technique for improving software system quality with higher number of test cases. To optimize the order of interactions being tested and to reduce the number of test suites generated best in terms of coverage, Combinatorial First Order Polynomial Coverage Based Prioritization (CFOP-CP) technique is proposed. For monitoring the order of interactions at time interval ' t ' and time interval ' t-1 ', First Order Polynomial (FOP) function is used. Therefore, FOP function predicts the faults interactions in software program efficiently which resulting in higher fault interaction prediction accuracy. After that, Coverage-based Test Suites Prioritization is performed to prioritize test suites best in terms of coverage and therefore provides global coverage for finding faults in software programs. Finally, Similarity-based Test Suite Selection is performed to reduce the number of test suites for detecting maximum number of faults in software programs which in turn helps for improving the software system quality. The CFOP-CP technique conducts the experimental works on parameters such as fault interaction prediction accuracy, testing cost and coverage rate. The experimental result shows that the CFOP-CP technique is able to improve the coverage rate for software fault detection and also reduces the testing cost for improving the software system quality when compared to state-of-the-art-works.

Keywords: Combinatorial Testing, Test Suite, Fault Interaction, Coverage, Test Cases, Prioritization

1. INTRODUCTION

Software fault localization and diagnosis is a significant step in the Combinatorial Testing (CT) for improving software system quality. In CT, the new test cases are created over time due to software modifications that increases the size of test suite. The increased test suites size may enhances the testing cost of software program. Therefore, test suite minimization techniques are required to reduce the testing cost of fault localization with higher number of test cases and to improve the software system quality.

Many research works has been developed for test suite minimization. For example, a combinatorial testing algorithm (comFIL) was designed in [1] for locating interactions that cause the system's defeat to acquire minimum target interactions in test suite for generating Combinatorial testing. The comFIL was more precise in fault localization. However, the number of test cases generated is higher. The combinatorial optimization technique was presented in [2] for reducing the number of test cases in configuration-aware structural testing. The combinatorial optimization technique prioritizes the test case based on the fault density. But, performance of combinatorial optimization technique was not sufficient for identifying more faults.

Cuckoo Search (CS) Algorithm was developed in [3] that utilize combinatorial optimization concepts to reduce the number of test cases. CS technique needs more computational time. A novel approach was intended in [4] to significantly reduce the count of interactions to be tested without significant loss of fault detection capability. But, time complexity for deriving interactions for combinatorial testing was not considered.

A novel method was introduced in [5] for performing similarity-based test suite reduction in the context of model-based testing. Though, the test suite reduction is not efficient for covering all the faults. The optimization of test cases through prioritization was performed in [6] using genetic algorithm. But, optimized test cases do not cover more number of faults. A fuzzy software quality evaluation model was designed in [7] depends on weighted mutation rate correction incompletion G1 combination weights for software quality evaluation.

Test suite prioritization was accomplished in [8] with the aid of cost based combinatorial interaction coverage to improve the rate at which faults are detected in relation to cost. Test suite prioritization was not efficient for improving the software system quality. A similarity-based test case prioritization technique was introduced in [9] to enhance the fault detection rate and to discover bugs in loops. However, number of mutation faults and the number of test cases influence the results of test case prioritization techniques. A family of similarity-based test case selection techniques was presented in [10] to discover best tradeoffs among the number of test cases to run and fault identification. But, total execution cost was higher.

Based on the above mentioned techniques and methods presented, Combinatorial First Order Polynomial Coverage Based Prioritization (CFOP-CP) technique is developed. The research objective of CFOP-CP technique is formulated as follows,

- To predict the faults interactions in software programs, First Order Polynomial function is used in CFOP-CP technique.
- To prioritize test suites best in terms of coverage, Coverage-based Prioritization is carried out in CFOP-CP technique.
- To reduce the number of test suites with higher number of test cases, Similarity-based Test Suite Selection is employed in CFOP-CP technique.

The rest of this paper is structured as follows. Section 2 explains Combinatorial First Order Polynomial Coverage Based Prioritization (CFOP-CP) technique with the assist of architecture diagram. Section 3 and Section 4 explains the experimental settings and details performance analysis with the aid of parameters. Section 5 describes the related works. Finally, Section 6 concludes this paper.

2. COMBINATORIAL FIRST ORDER POLYNOMIAL

COVERAGE BASED PRIORITIZATION IDENTIFY TECHNIQUE

CT focuses on discovering faults that arise due to interaction of values of a small number of input parameters. Recently, most of research works has been developed for test suite minimization for CT. However, existing test suite minimization techniques does not covers the maximum number of test cases for identifying the more faults in software programs. In order to overcome such limitation, Combinatorial First Order Polynomial Coverage Based Prioritization (CFOP-CP) technique is designed. The main objective of CFOP-CP technique is to optimize the order of interactions being tested and to reduce the number of test suites generated best in terms of coverage for detecting the more number of faults in software programs. For monitoring the order of interactions and to identify the fault interactions in software program, CFOP-CP technique used First Order Polynomial (FOP) function. The FOP function efficiently predicts the number of faults interaction in software program with the aid of evaluation error.

For prioritizing the test suites best in terms of coverage (i.e. maximum number of test cases) and providing the global coverage for detecting the maximum number of faults in software programs, CFOP-CP technique used Coverage-based Prioritization process. For reducing the number of test suites best in terms of coverage, CFOP-CP technique performs Similarity-based Test Suite Selection process. The overall architecture diagram of Combinatorial First Order Polynomial Coverage Based Prioritization technique is shown in below Figure 1.

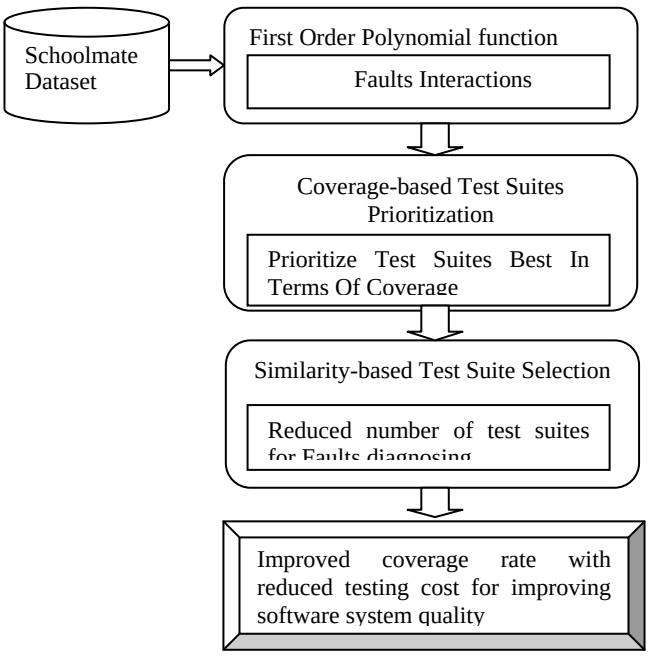

Figure 1 Architecture Diagram of Combinatorial Bayesian Order Polynomial Coverage-based Prioritization Technique

As shown in Figure 1, CFOP-CP technique initially takes Schoolmate Dataset as input. Then, CFOP-CP technique used FOP function for screening the order of interactions and finding the fault interactions in software program. By using FOP function, CFOP-CP technique efficiently identifies the fault interaction. This in turn helps for CFOP-CP technique for improving the fault interaction prediction accuracy. After that, Coverage-based Test Suites Prioritization is accomplished in order to prioritize test suites best in terms of coverage which resulting in enhanced coverage rate. Finally, Similarity-based Test Suite Selection is performed in order to obtain the reduced number of test suites to find the faults in identified faults interactions which in reduces the number of count of interactions to be tested. As a result, CFOP-CP technique significantly reduces the testing cost of software program.

2.1 First Order Polynomial Function

The CFOP-CP technique used first Order Polynomial (FOP) Function to identify the faults interactions that exist in the source code, thereby reducing the count of interactions to be tested. With the aid of FOP Function, CFOP-CP technique finds possible interactions that may cause the failure to occur or identified at an earlier stage, i.e. even before the occurrence of failures. Thus, CFOP-CP technique tests only the identified fault interactions rather than testing all the N interactions in software programs. This in turn assists for significant reduction in count of interactions to be tested and therefore reduces testing costs. Hence, software testers can easily examine and locate the factors relevant to defects of system more accurately, thus making the process of software testing and debugging easier and more effectual.

The FOP Function is a simple, non-trivial and time series model where the monitoring of interactions $^{M_{I_N}}$ in software program at the time t and v_t is mathematically represented as,

$$M_{I_N} = I_t + v_t \quad \text{Where} \quad v_t \sim N[o, V_t] \qquad (1)$$

From the equation (1), I_t is the order of the interactions at time t and v_t is the observational error. The time evolution of the interactions is then modelled as a locally constant mean with evolution error u_t which mathematically represented as,

$$I_t = I_{t-1} + u_t \qquad (2)$$

From the equation (3), u_t is fault interaction. Therefore, the forecasting of fault interactions form the software program at time interval '*t*' and time interval '*t-1*' is mathematically formulated as,

$$(I_t | I_{t-1}) \sim I_N[M_{I_N}, FI_N] \qquad (3)$$

From the equation (3), I_t denotes the order of the interactions at time t and I_{t-1} represents order of the interactions at time *t-1* whereas $^{M_{I_N}}$ indicates the monitoring of interactions in software program. Here, $^{F_{I_N}}$ is the number of identified fault interactions form the total interactions of software program using evaluation error. Therefore, CFOP-CP technique efficiently identifies the fault interactions form the *N* interactions of software program which resulting in improved fault interaction prediction accuracy. This in turn helps for predicting the future fault interactions in software program. After detecting the fault interactions in software program, test suite prioritization and selection process is accomplished in order to optimize the number of test suites best in terms of

coverage for software program evaluation which is detailed explained in forthcoming sections.

2.2 Coverage-Based Prioritization

In CFOP-CP technique, test suite prioritization is performed for prioritizing test suite in order that increases their effectiveness in meeting some performance goals for improving the software quality. The test suite is prioritized based on coverage criterion. The prioritization based on coverage is depends on the hypothesis that early maximization of coverage could lead to early detection of faults. The coverage-based test suite prioritization prioritizes test suites based on their total coverage of test cases for identifying more faults. The process of coverage-based test suite prioritization is shown in below Figure 2.

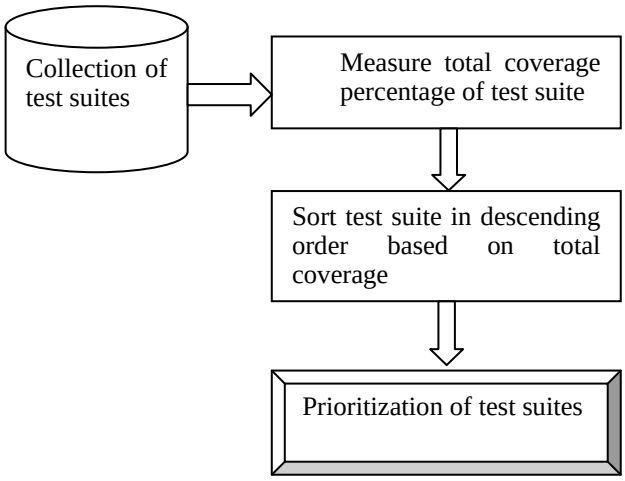

Figure 2 Process of Coverage-Based Test Suite Prioritization

The test suites prioritization problem in CFOP-CP technique is mathematically formulated as follows,

$$Test\ \ Suite\ Prioritization = (TS, N, f) \qquad (4)$$

From the equation (4), *TS* is a test suite. Here, *N* represents all the test suites obtained through performing the tests and f is a function form *N* to the real numbers. Therefore, the problem is to find test suite form the set of all test suites is mathematically represented as follow,

$$\pi \in N \text{ Such that } \forall \pi' \in N,\ f(\pi) \geq f(\pi') \qquad (5)$$

Form these equations (4) and (5), possible prioritizations of TS are referred to as N in which f is a function to measure the orderings. The test suite prioritization is based on any criterion. In CFOP-CP technique, test suite prioritization based on interaction faults coverage.

Let us assume the function for the test suite prioritization problem through interaction faults coverage. Given a test suite *TS*, *N* is the set of all test suites acquired through performing the ordering of tests. Each permutation is referred to as $\pi_i \in N$ and an individual permutation contains n test cases $\pi_i = \pi_{i1,}\pi_{i2,\ldots,}\pi_{in}$. A function $tCoverage(\pi_{ik})$ computes the set of covered *t*-tuples in a test π_{ik}. Hence, total coverage percentage of test suite (i.e. the test suites that covers maximum faults in software program) is measured for the prioritizing test suites which is mathematically formulated as,

$$f(\pi_i) = \sum_{j=0}^{n} \left| \bigcup_{k=0}^{j} tCoverage(\pi_{ik}) \right| \qquad (6)$$

From the equation (6), the total coverage percentage of test suite is measured to find the more faults in software programs at the earlier stage. The algorithmic process of coverage-based test suite prioritization is shown in below,

// Coverage-Based Test Suite Prioritization
Input: Number Of Test Suites with n test cases
Output: Prioritized Test Suites
Step 1: Begin
Step 2: **For** each test suite
Step 3: Compute total coverage percentage using (6)
Step 4: **End for**
Step 5: Sort test suite in descending order based on the value of total coverage percentage for prioritization
Step 6: Outputs the prioritized test suites list
Step 7: End

Algorithm 1 Coverage-Based Test Suite Prioritization

As shown in Algorithm 1, During the Coverage-Based Test Suite Prioritization process, the test suite with the highest coverage percentage is placed to an initially empty solution, followed by next highest percentage is included, until all test suites have been added. This in turn helps for achieving global coverage to reduce the testing cost and to find the more faults of software programs at an earlier stage for enhancing the software system quality.

2.3 Similarity-Based Test Suite Selection

In CFOP-CP technique, similarity-based test suite selection is performed to select the reduced number of test suites which covers the maximum number of test cases for improving the quality of software program evaluation. This similarity based test suite selection process reduces the number of redundant test cases with the aid of measured similarity value. The process of similarity-based test suite selection is shown in below Figure 3,

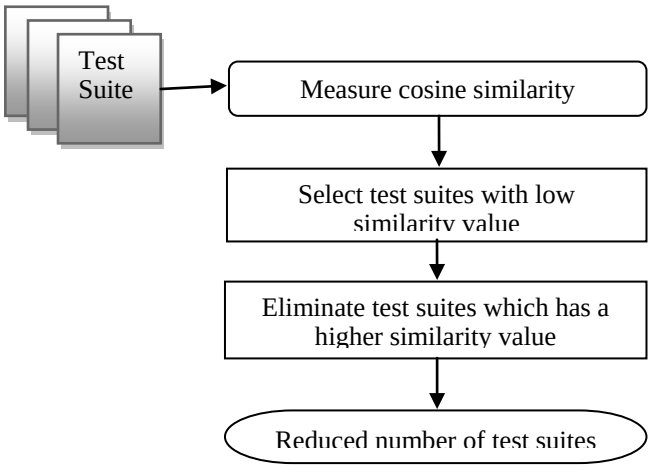

Figure 3 Process of Similarity-Based Test Suite Selection

As shown in Figure 3, similarity-based test suite selection process initially evaluates the cosine similarity value for prioritized test suites. Next, similarity-based test suite selection process eliminates test suites which have a higher similarity value and then choose test suites with low similarity value for efficient software program evaluation which resulting in reduced number of test suites.

The maximum number of test cases coverage covered by test suite t_1 and t_2 are $X:<x_1,x_2,x_3,\ldots,x_n>$ and $Y:<y_1,y_2,y_3,\ldots,y_n>$ respectively. Therefore, similarity between test suites is measured by using cosine similarity which is mathematically formulated as,

$$Similarity(t_1,t_2) = \frac{x^t.y}{\|x\|\,\|y\|} \qquad (7)$$

From the equation (7), similarity between test suites is evaluated for test suite selection. Here, x^t is a transposition of vector x and $\|x\|$ is the Euclidean norm of vector x. The Euclidean norm of vector x can be calculated according to $\sqrt{x_1^2 + x_2^2 + \ldots + x_n^n}$. Similarly, $\|y\|$ is the Euclidean norm of vector y.

With the aid of measured similarity value of test suites, CFOP-CP chooses the test suites with lower similarity value. Therefore, CFOP-CP technique obtains the reduced number of test suites for software program evaluation. The algorithmic process of Similarity-based Test Suite Selection for software program evaluation is shown in below,

// Similarity-based Test Suite Selection Algorithm

Input: Prioritized Test Suites

Output: Test suites Reduction (Select The Test Suites That Covers Most Number Of Test Cases For Software Validation)

Step 1:Begin

Step 2: For each prioritized test suites

Step 3: Measure similarity between test suites using (7)

Step 4: Choose the test suites which has a low similarity value

Step 5: Remove test suites which has a higher similarity value

Step 6: End for

Step 7: End

Algorithm 2 Similarity-Based Test Suite Selection

As shown in algorithm 2, Based on the similarity value between each pair of test cases, the test suites with biggest similarity value are eliminated and test suites with lower similarity value is selected. This in turn provides reduced number of test suites with more number of test cases for finding maximum number of faults in software program which resulting in improved software system quality.

The reduced number of test suites is obtained from Similarity-based Test Suite Selection Algorithm is applied on identified fault interactions for detecting faults. Therefore, CFOP-CP technique reduces the count of interactions to be tested which resulting in reduced testing costs.

3. EXPERIMENTAL SETTINGS

The proposed Combinatorial First Order Polynomial Coverage Based Prioritization (CFOP-CP) technique is implemented in Java Language by using schoolmate data set. The CFOP-CP technique employed schoolmate data set for identifying faults in software programs to improve software system quality. This schoolmate data set consists of many PHP program. The performance of CFOP-CP technique is measured in terms of fault interaction prediction accuracy, coverage rate and testing cost.

4. RESULT AND DISCUSSIONS

In this section, the result analysis of CFOP-CP technique is estimated. The effectiveness CFOP-CP technique is compared against with two methods namely combinatorial testing algorithm (comFIL) [1] and combinatorial optimization technique [2] respectively. The efficiency of CFOP-CP technique is evaluated along with the following metrics with the help of tables and graphs.

4.1 Measurement of Fault Interaction Prediction Accuracy

The fault interaction prediction accuracy is defined as the ratio of number of correctly identified fault interactions to the total number of interactions taken. The fault interaction prediction accuracy is measured in terms of percentages (%) and mathematically expressed as,

$$\text{Fault Interaction Prediction Accuracy} = \frac{\text{identified fault interactions}}{\text{total number of interactions}} * 100 \qquad (8)$$

From the equation (8), fault interaction prediction accuracy is obtained. While the fault interaction prediction accuracy is higher, the method is said to be more efficient.

Table 1 Tabulation for Fault Interaction Prediction Accuracy

Number Of Interactions	Fault Interaction Prediction Accuracy (%)		
	comFIL	Combinatorial Optimization Technique	CFOP-CP Technique
10	55.11	66.87	79.25
20	56.93	68.42	80.93
30	57.18	71.26	82.37
40	60.79	72.39	84.78
50	61.26	75.14	85.91
60	63.74	76.92	86.85
70	64.52	78.26	88.91
80	67.19	82.44	91.27
90	69.14	83.67	92.88
100	72.35	85.93	94.67

Table 1 demonstrates the tabulation for fault interaction prediction accuracy based on different number of interactions taken in the range of 10-100. The CFOP-CP technique considers the framework with different number of interactions for conducting experimental works using Java Language. From the table, it descriptive that the fault interaction prediction using proposed CFOP-CP technique is higher when compared to other existing works namely comFIL [1] and combinatorial optimization technique [2].

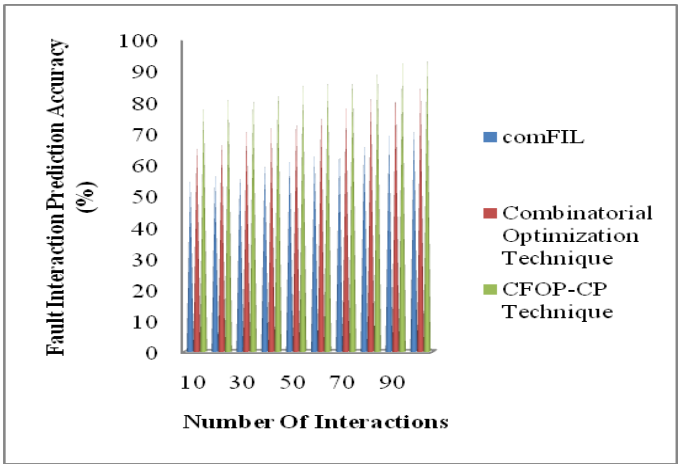

Figure 4 Measure of Fault Interaction Prediction Accuracy

Figure 4 depicts the impact of fault interaction prediction accuracy versus different number of interactions using three methods. As exposed in figure, the proposed CFOP-CP technique is provides better fault interaction prediction accuracy for improving the software system quality when compared to existing two works namely comFIL [1] and combinatorial optimization technique [2] . In addition, while increasing the number of interactions, the fault interaction prediction accuracy is also gets increased using all three methods. But comparatively, the fault interaction prediction accuracy using proposed CFOP-CP technique is higher. This is due to application of FOP function in CFOP-CP technique where it monitors the order of interactions and efficient prediction in fault interactions with the aid of evaluation error. This in turn helps for improving the fault interaction prediction accuracy in a significant manner. Therefore, the CFOP-CP technique improves fault interaction prediction accuracy by 39 % when compared to comFIL [1] and 14 % when compared to combinatorial optimization technique [2] respectively.

4.2 Measurement of Coverage Rate

The coverage measures the rate at which a maximum number of faults covered by a selected test suites form the total number of test suites for increasing the software system quality. The coverage rate is measured in terms of percentages (%) and mathematically formulated as,

$$\text{Coverage Rate} = \frac{\text{total number of test suites-selected test suites for identifying more faults}}{\text{total number of test suites}} * 100 \quad (9)$$

From the equation (9), coverage rate of test suites for identifying the more number of faults in given software program is obtained. While the coverage rate is higher, the method is said to be more efficient.

Table 2 Tabulation for Coverage Rate

Number Of Test suites	Coverage Rate (%)		
	comFIL	Combinatorial Optimization Technique	CFOP-CP Technique
10	52.36	61.78	70.25
20	53.98	63.87	73.58
30	57.89	66.15	75.19
40	58.14	67.05	76.97
50	60.78	69.34	78.36
60	61.02	71.71	80.15
70	63.25	72.56	84.59
80	64.87	74.95	85.94
90	65.90	78.97	87.16
100	67.23	79.99	88.97

Table 2 depicts the tabulation of coverage rate for discovering the maximum number of faults based on different number of test suites taken in the range of 10-100. From the table, it illustrative that the coverage rate using proposed CFOP-CP technique is higher when compared to other existing works namely comFIL [1] and combinatorial optimization technique [2].

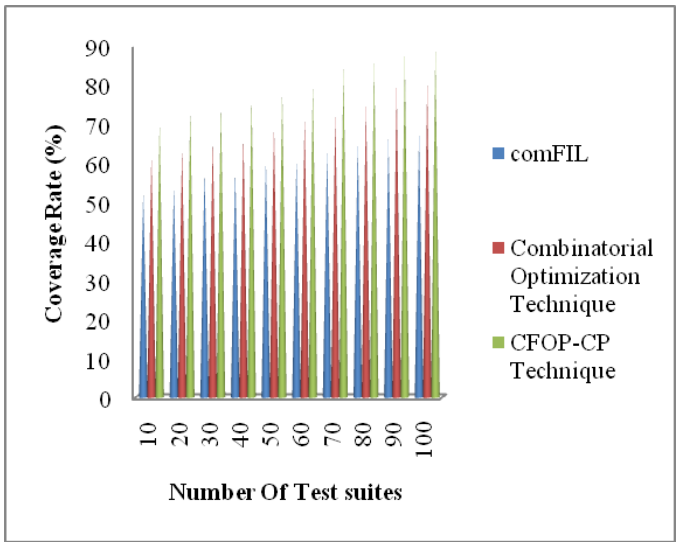

Figure 5 Measure of Coverage Rate

Figure 5 portrays the impact of coverage rate of faults versus different number of test suites using three methods. As illustrated in figure, the proposed CFOP-CP technique is provides better coverage rate for finding the more faults in software program when compared to existing two works namely comFIL [1] and combinatorial optimization technique [2] . As well, while increasing the number of test suite, the coverage rate is also gets increased using all three methods. But comparatively, the coverage rate using proposed CFOP-CP technique is higher. This is because of application of Coverage-Based Test Suite Prioritization and Similarity-Based Test Suite Selection algorithms in CFOP-CP technique. With the support of these two algorithmic processes, CFOP-CP technique attains the reduced number of test suites generated best in terms of coverage with higher number of test cases for detecting the more faults. This in turn assists for improving the coverage rate in an efficient manner. As a result, the CFOP-CP technique improves coverage rate by 32 % when compared to

comFIL [1] and 14 % when compared to combinatorial optimization technique [2] respectively.

4.3 Measurement of Testing Cost

The testing cost measures the amount of time taken by selected test suites for finding the maximum number of faults in software program. The test cost is measured in terms of milliseconds (ms) and mathematically represented as below,

$$Test\ Cost = N * T_t \qquad (10)$$

From the equation (10), testing cost for identifying the more number of faults in given software program is obtained where N is the total number of test suites and T_t is time taken for identifying faults by one test suite. While the testing cost is lower, the method is said to be more efficient.

Table 3 Tabulation for Testing Cost

Number Of Test suites	Testing Cost (ms)		
	comFIL	Combinatorial Optimization Technique	CFOP-CP Technique
10	36.9	29.7	23.5
20	38.2	33.5	28.6
30	44.6	38.9	32.2
40	47.3	41.2	35.9
50	51.7	45.4	41.8
60	55.8	51.1	45.7
70	62.2	55.8	51.8
80	65.1	57.7	55.3
90	69.9	62.5	59.7
100	72.3	66.2	62.4

The comparative result analysis of testing cost with respect to different number of test suites in the range of 10-100 is presented in Table 3. From the table, it expressive that the testing cost using proposed CFOP-CP technique is lower when compared to other existing works namely comFIL [1] and combinatorial optimization technique [2].

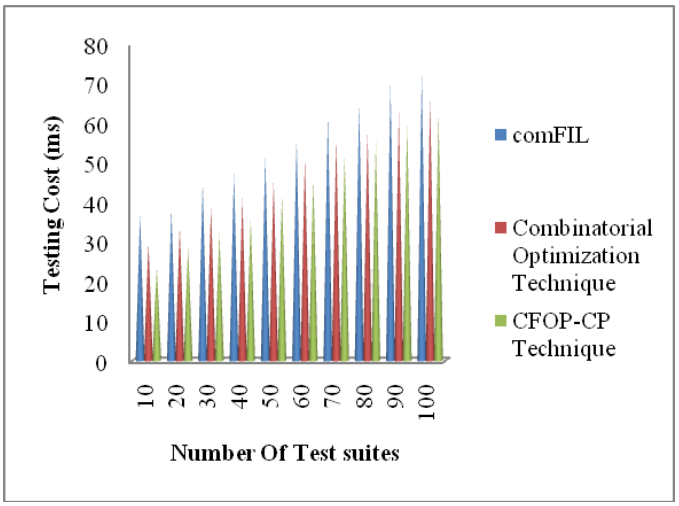

Figure 6 Measure of Testing Cost

Figure 6 describes the impact of testing cost versus diverse number of test suites using three methods. As demonstrated in figure, the proposed CFOP-CP technique provides better testing cost for improving software system quality when compared to existing two works namely comFIL [1] and combinatorial optimization technique [2] . Also, while increasing the number of test suite, the testing cost is also gets increased using all three methods. But comparatively, the testing cost using proposed CFOP-CP technique is lower. This is because of application of FOP function in CFOP-CP technique. With the support of FOP function, CFOP-CP technique monitors the order of interactions and predicts the fault interactions in software program by using evaluation error. Hence, CFOP-CP technique tests only the discovered fault interactions rather than testing all the N interactions in software programs. This in turn reduces the count of interactions to be tested and which resulting in reduced testing costs. Thus, the CFOP-CP technique reduces testing cost by 21 % when compared to comFIL [1] and 11% when compared to combinatorial optimization technique [2] respectively.

5. RELATED WORKS

An Analysis of Test Suite Minimization Techniques was performed in [11] for improving the testing process of fault detection and achieving all the testing requirements. An Improved History-Based Test Prioritization Technique was designed in [12] using code coverage to detect fault faster. However, more number of faults is not identified. A variable strength combinatorial strategy was developed in [13] for performing testing which covers the needed valid combinations of parameters including mutual interactions in a concurrent program. But, this strategy decreases the capability of fault detection.

An efficient algorithm pair wise test set generator using genetic algorithm (PTSG-GA) was intended in [14] for generating test set for pair-wise testing. The PTSG-GA is not designed to incorporate constraint handling feature. Genetic algorithm was designed in [15] for constrained Combinatorial Interaction Testing in which pair wise testing is performed for subjects with a small number of constraints. However, impact of reduced test suites for the fault detection was not sufficient. Quality-Aware Service Selection for Service-Based Systems was presented in [16] to discover a solution which attains the service-based systems owner's optimisation goal while achieving all quality constraints for the service-based systems. But, the increase in number of quality constraints affects its performance to find a solution.

A novel algorithm called Pairwise with constraints, Order and Weight (PROW) was employed in [17] for handling constraints and prioritization for pairwise coverage. The different techniques designed for software fault-localization methods was analyzed in [18] for increasing the quality of software system. Fault Localization Method was intended in [19] based on dependencies analysis of program structure for enhancing the accuracy of fault localization. But the accuracy of fault localization was not at required level. An iterative fault localization process was performed in [20] to select test cases for fault localization and to find many faults in the program. However, effective fault localization was not performed.

6. CONCLUSION

An efficient Combinatorial First Order Polynomial Coverage Based Prioritization (CFOP-CP) technique is designed with objective of optimizing the order of interactions being tested and reducing the number of test suites generated best in terms of coverage. Initially, CFOP-CP technique employed FOP function for monitoring the order of interactions and predicting the faults interactions in software

program efficiently. Therefore, CFOP-CP technique attains the higher fault interaction prediction accuracy. Next, coverage-based test suites prioritization is performed CFOP-CP technique with the aim of prioritizing test suites best in terms of coverage and therefore improves the coverage rate for detecting the more number of faults in software programs. At last, Similarity-based Test Suite Selection is performed in CFOP-CP technique in order to minimize the number of test suites with higher number of test cases for discovering maximum number of faults in software programs. By using reduced number of test suites, CBOP-CP technique tests only the identified fault interactions rather than testing all the N interactions in software programs. This in turn assists for reducing the count of interactions to be tested and which resulting in reduced testing costs for improving the software system quality. The efficiency of CBOP-CP technique is test with the metrics such as fault interaction prediction accuracy, coverage rate and testing cost. With the experiments conducted for CBOP-CP technique, it is observed that the coverage rate for improving software system quality provided more accurate results as compared to state-of-the-art works. The experimental results demonstrate that CBOP-CP technique is provides better performance with an improvement of coverage rate and also reduces the testing cost when compared to the state-of-the-art works.

REFERENCES

[1] Wei Zheng, Xiaoxue Wu, Desheng Hu, and Qihai Zhu, "Locating Minimal Fault Interaction in Combinatorial Testing", Hindawi Publishing Corporation, Advances in Software Engineering, Volume 2016, Article ID 2409521, Pages 1-10, 2016

[2] Bestoun S. Ahmed, "Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing", Engineering Science and Technology, an International Journal, Elsevier, Volume 19, Issue 2, Pages 737–753, June 2016

[3] Bestoun S. Ahmed, Taib Sh. Abdulsamad and Moayad Y. Potrus, "Achievement of Minimized Combinatorial Test Suite for Configuration-Aware Software Functional Testing Using the Cuckoo Search Algorithm", Information and Software Technology, Volume 66, Pages 13-29, Oct 2015

[4] Sangeeta Sabharwal, Manuj Aggarwal, "A novel approach for deriving interactions for combinatorial testing", Engineering Science and Technology, an International Journal, Elsevier, Volume 20, Pages 59–71, 2017

[5] Ana Emilia Victor Barbosa Coutinho, Emanuela Gadelha Cartaxo, Patricia Duarte de Lima Machado, "Analysis of distance functions for similarity-based test suite reduction in the context of model-based testing", Software Quality Journal, Volume 24, Issue 2, Pages 407–445, June 2016

[6] T. Prem Jacob and T. Ravi, "Optimization of Test Cases by Prioritization", Journal of Computer Science, Volume 9, Issue 8, Pages 972-980, 2013

[7] Chuanyang Ruan and Jianhui Yang, "Software Quality Evaluation Model Based on Weighted Mutation Rate Correction Incompletion G1 Combination Weights", Hindawi Publishing Corporation, Mathematical Problems in Engineering, Volume 2014, Article ID 541292, Pages 1-9, 2014

[8] Renée C. Bryce, Sreedevi Sampath, Jan B. Pedersen, Schuyler Manchester, "Test suite prioritization by cost-based combinatorial interaction coverage", International Journal of System Assurance Engineering and Management, Springer, June 2011, Volume 2, Issue 2, Pages 126–134

[9] Chunrong Fang, Zhenyu Chen, Zhihong Zhao, "Similarity-based test case prioritization using ordered sequences of program entities", Software Quality Control, Volume 22, Issue 2, Pages 335–361, June 2014

[10] Hadi Hemmati, Andrea Arcuri, Lionel Briand, "Achieving Scalable Model-Based Testing Through Test Case Diversity", ACM Transactions on Software Engineering and Methodology, Volume 22, Issue 1, Pages 1-42, February 2013

[11] Shilpi Singh, Raj Shree, "An Analysis of Test Suite Minimization Techniques", international Journal of Engineering Sciences and Research Technology, Volume 5, Pages 252-260, 2016

[12] Avinash Gupta, Nayneesh Mishra, Aprna Tripathi, Manu Vardhan, Dharmender Singh Kushwaha, "An Improved History-Based Test Prioritization Technique Using Code Coverage", Advanced Computer and Communication Engineering Technology, Pages 437-448, 2014

[13] Xiaofang QI , Junhe, Peng Wang, Huayang Zhou, "Variable strength combinatorial testing of concurrent programs", Springer, Frontiers of Computer Science, Volume 10, Issue 4, Pages 631–643, August 2016

[14] Sangeeta Sabharwal, Priti Bansal, Nitish Mittal, Shreya Malik, "Construction of Mixed Covering Arrays for Pair-wise Testing Using Probabilistic Approach in Genetic Algorithm", Arabian Journal for Science and Engineering, Springer, Volume 41, Issue 8, Pages 2821–2835, August 2016

[15] Justyna Petke, Myra B. Cohen, Mark Harman, and Shin Yoo, "Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection", IEEE Transactions on Software Engineering, Volume 41, Issue 9, Pages 901 – 924, 2015

[16] Qiang He, Jun Yan, Hai Jin and Yun Yang, "Quality-Aware Service Selection for Service-Based Systems Based on Iterative Multi-Attribute Combinatorial Auction", IEEE Transactions On Software Engineering, Volume 40, Issue 2, Pages 192-215, February 2014

[17] Beatriz Perez Lamancha, Macario Polo, Mario Piattini, "PROW: A Pairwise algorithm with constraints, Order and Weight", The Journal of Systems and Software, Elsevier, Volume 99, Pages 1–19, 2015

[18] Pragya Agarwal, Arun Prakash Agrawal, "Fault-Localization Techniques for Software Systems: A Literature Review", ACM SIGSOFT Software Engineering, Volume 39, Issue 5, Pages 1-8, September 2014

[19] Hui He, Lei Zhao, Qiao Li, Weizhe Zhang, Dongmin Gao and Yongtan Liu, "Fault Localization Method of Software Defects based on Dependencies Analysis of Program Structure", International Journal of Security and Its Applications, Volume 7, Issue 3, Pages 413- 422, May 2013

[20] Xiaobing Sun, Xin Peng Bin, LiBixin Li, Wanzhi Wen, "IPSETFUL: an iterative process of selecting test cases for effective fault localization by exploring concept lattice of program spectra", Frontiers of Computer Science, Springer, Volume 10, Issue 5, Pages 812–831, October 2016