Contribution-based Test Case Reduction Strategy for Mutation-based Fault Localization

Haifeng Wang†, Kun Yang†*, Xiangnan Zhao†, Yuchen Cui†, Weiwei Wang‡
†*Center for Advanced Metering Infrastructure*, *National Institute of Metrology*, China
‡*College of Information Science and Technology*, *Beijing University of Chemical Technology*, China
Email: wanghaifeng@nim.ac.cn

***Abstract*—Fault localization is one of the most expensive steps during software debugging. Mutation-based Fault Localization (MBFL) is a promising technique but with a high computational cost since the large mutation execution on mutation analysis. Previous studies mainly reduce the cost of decreasing the number of mutants and optimizing the process execution, such kinds of strategies have shown promising results. However, reducing the cost of MBFL by decreasing the number of test cases is also effective. In this paper, we propose a Contribution-Based Test Case Reduction (CBTCR) strategy for improving the efficiency of MBFL. CBTCR first measures the contribution value of each test case and then selects the test cases according to the value. Then it takes the reduced test suite for executing the mutants. We evaluate CBTCR on 383 real software faults from the Defects4J benchmark. The results show that CBTCR outperforms the other MBFL test case reduction strategies (e.g., FTMES, IETCR) in terms of the Top-N and MAP metrics. Also, CBTCR can significantly reduce 85.43% of the cost on average while maintaining a similar accuracy to original MBFL techniques.**

***Keywords*—software debugging; fault localization; mutation-based fault localization; test case reduction**

I. Introduction

Software debugging is an expensive and difficult process that costs developers a large amount of time and effort [1]. The first step in debugging, called fault localization (FL), is to identify the root cause of observed failures. FL is a complicated and time-consuming task, as it is important in software maintenance and evolution. To alleviate the human effort in localizing a fault, various automated FL techniques have been proposed, e.g., spectrum-based [2] and mutation-based [3], [4] techniques.

Spectrum-based fault localization (SBFL), is one of the most widely studied techniques. SBFL considers the binary coverage of the program elements but with inherently limited fault localization accuracy. Recent studies have shown that mutation-based fault localization (MBFL) techniques can help improve the performance of fault localization and achieve a higher fault localization accuracy than the state-of-the-art SBFL techniques [3].

MBFL is based on mutation analysis and works by making syntactic changes to the program. Although the studies have shown that MBFL aims the localize accuracy improvement of suspicious elements list for fault software debugging, the computational cost is high for such techniques. Hence, researchers investigate optimization strategies for MBFL techniques, which can be divided into three groups: (1) reduce the mutants [1], [3], [5]; (2) reduce the test cases [6], [7]; and (3) optimize the execution [8]. However, other opportunities exist for optimization.

In this study, a novel test case reduction strategy named CBTCR (Contribution-Based Test Case Reduction) is proposed aiming at optimizing the mutation execution. CBTCR keeps both failed and passed test cases like IETCR [7] but is different from FTMES [6], which only employs the failed ones. In detail, CBTCR uses test cover and suspiciousness of SBFL to measure the contribution value of test cases for MBFL. Next, the test cases are executed with higher values and avoid the execution of the rest ones.

The contributions of this paper are as follows:

- This paper proposes a contribution-based test case reduction (CBTCR) strategy to reduce the mutation execution cost of MBFL.
- This paper reports an empirical study on 383 real-fault programs from Defects4J, and the results show that MBFL execution cost could be remarkably reduced while adopting CBTCR. At the same time, the fault localization accuracy is kept almost the same as the original MBFL.
- The scripts and dataset used in this paper are all available in the GitHub repository[1] to facilitate the replication of our study and evaluation of future work.

The rest of this paper is organized as follows. Section II summarizes the background and related work. Section III describes the details of our approach. Section IV illustrates the experimental setup and analyzes our experimental results. Finally, Section V summarizes our study with potential future work.

II. Background and Related Work

A. Mutation-Based Fault Localization

Mutation-based fault localization is a well-studied technique that is based on mutation analysis. In mutation analysis, mutants are used to evaluate the quality of test cases based on their ability to distinguish the mutants' behavior from that of the original program [3]. Mutants can be evaluated by suspiciousness using the MBFL formulas. MBFL works

DOI reference number: 10.18293/SEKE2023-180

[1]https://github.com/hfwiz/CBTCR

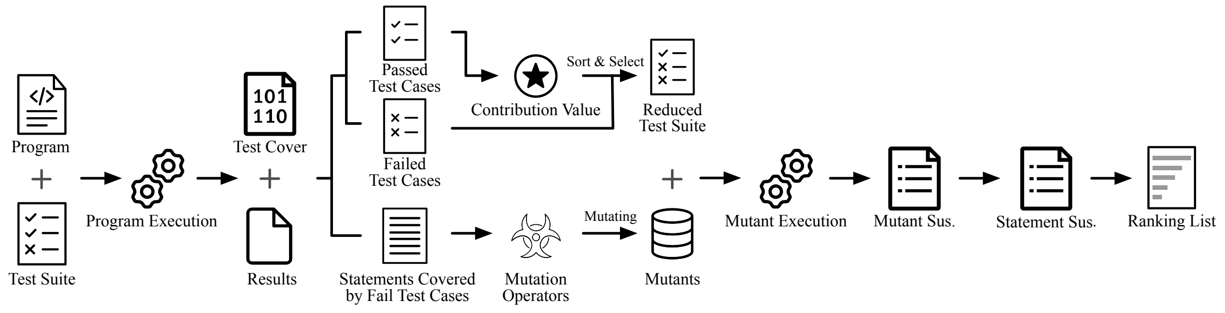

Fig. 1. The CBTCR workflow

based on the assumption that mutants killed mostly by failed test cases have a connection with program faults. Recent studies [3] also demonstrated that MBFL could significantly outperform other types of fault localization techniques.

B. MBFL Reduction Strategies

MBFL achieves high accuracy of fault localization but faces huge computational costs on execution. Various strategies have been presented for reducing the cost of MBFL, which can be divided into three categories: (1) reduce the mutants [1], [3]–[5], [9]. SELECTIVE [5] selects the ”sufficient” mutation operators for generating mutants. SMBFL [1] reduces the mutants by examining only the statements in the dynamic program slice. In another aspect, SAMPLING [3], SOME [9], and WSOME [4] are reducing the cost by decreasing the mutants from the mutant set. (2) reduce the test cases [6]. FTMES [6] uses only the set of failed test cases to execute mutants while avoiding the execution of passed test cases. IETCR [7] measures the information of test cases using entropy and selects a proportion of them. (3) optimize the execution [8]. DMES [8] contains two kinds of optimizations, i.e., mutation execution optimization and test case execution optimization.

III. Our Method

A. Contribution Value of Tests

In software engineering, a good quality software it is essential to rely on good test cases, which are more powerful to discover bugs in the programs. It is the classic objective of testing and a failed test is more powerful than a passed test in that it can help find defects. In mutation-based fault localization, the test suite has been run first for obtaining elements covered by fail tests, so it is easy to distinguish the passed and failed test cases before executing mutants. However, how to distinguish the good and bad of the passed test cases is unclear.

To fill this gap, in this paper, we define **Contribution value** to measure the possible fault localization effectiveness for MBFL. Here, we will clarify the meaning of basic terms and notations used in this paper, including programs, tests, coverage, and suspiciousness.

Definition 1: Test cover. Given a program $P = \{s_1, s_2, ..., s_n\}$ with n elements, a statement $s \in P$, a test $t \in T = \{t_1, t_2, ..., t_m\}$ be a set of all tests. If t executes a statement $s \in P$, it is said that $t\ covers\ s$, which is formalized by:

$$Cover(t, s) = \begin{cases} 1(\text{true}), & \text{if } t \text{ covers } s \\ 0(\text{false}), & \text{otherwise,} \end{cases} \tag{1}$$

By definition, a test cover concisely represents whether the statement s has executed the test t, which is the original of the spectrum-based fault localization.

For a statement s, the suspiciousness of s using SBFL formulas is denoted as $Sus(s)$ (the SBFL formula we called the contribution formula, Dstar [2] is used in this paper). Then we use **Contribution value** to quantitatively measure the contribution of a test t for locating the fault in program P. The **Contribution value**, simply the $cValue$, can be formalized as follows:

$$cValue(t, P) = \sum_{i=1}^{n} Cover\left(t, s_i\right) \times Sus\left(s_i\right) \tag{2}$$

for a test $t \in T$ and statements $s_1, s_2, \ldots, s_n \in P$.

In other words, the coverage of tests and the suspiciousness of statements imply the quantitative quality of the test t for MBFL. If a test t has a larger $cValue$ indicates that t can cover more statements with greater suspiciousness, which suggests that t is more valuable for MBFL.

B. The Workflow of CBTCR

Fig. 1 shows the workflow of our proposed CBTCR. At first, CBTCR executes the tests against the program to obtain test cover and results. Then, the tests are divided into passed and failed sets. In addition, the statements covered by failed test cases are obtained for generating mutants by seeding mutation operators. Next, CBTCR works by calculating the contribution value of passed test cases. After sorting and selecting test cases with higher contribution values, a reduced test suite is generated by combining passed and failed test cases. Finally, the reduced test suite is executed on the mutants to get killing information. The mutants' suspiciousness are calculated and the maximum one is is assigned to the statement. The MBFL sorts all statements in descending order based on their suspiciousness and returns a ranking list of all program elements.

TABLE I
Top-N AND *MAP* OF CBTCR AND OTHER STRATEGIES ON MBFL TECHNIQUES AT DIFFERENT SAMPLING RATIOS

Technique	**Strategy**	**Sampling ratio =**											
		10%				**20%**				**30%**			
		Top-1	**Top-3**	**Top-5**	**MAP**	**Top-1**	**Top-3**	**Top-5**	**MAP**	**Top-1**	**Top-3**	**Top-5**	**MAP**
MBFL	RAND	20	51	103	0.1792	29	75	117	0.2546	27	87	116	0.2364
	IETCR	10	53	93	0.1575	24	69	103	0.2228	32	78	113	0.2646
	FTMES	43	88	120	0.2532	43	88	120	0.2532	43	88	120	0.2532
	CBTCR	33	97	122	0.2554	38	103	131	0.2997	40	107	133	0.3163
	Original	41	108	137	0.3181				-				
MCBFL	RAND	44	106	134	0.2942	51	106	145	0.3357	47	106	139	0.3173
	IETCR	36	101	129	0.2808	43	105	133	0.3098	51	104	134	0.3309
	FTMES	46	92	131	0.2851	46	92	131	0.2851	46	92	131	0.2851
	CBTCR	58	115	138	0.3600	61	116	146	0.3777	62	116	152	0.3818
	Original	61	113	151	0.3789				-				

IV. Experimental Analysis

To investigate the performance of CBTCR, we perform experiments to address the following three research questions:

- **RQ1:** How does CBTCR perform compared to different test case reduction strategies (i.e., FTMES, IETCR) and the original MBFL techniques on fault localization effectiveness?
- **RQ2:** How much execution cost does CBTCR need when comparing other test case reduction strategies and original MBFL techniques?

RQ1 compares the effects of different test case reduction strategies for MBFL techniques and the performance of such techniques using CBTCR. **RQ2** is designed to evaluate the efficiency of MBFL adopting different reduction strategies.

A. Experimental setup

TABLE II
Subject programs

Project	**Versions (used)**	**kLOC**	**Average of Mutants**	**Average of $\|T_f\|$**	**Average of $\|T_p\|$**
Lang	65 (64)	18.5	1775	3.1	172.8
Chart	26 (21)	85.1	2986	2.0	194.2
Time	27 (16)	27.3	8326	103.1	3651.0
Math	106 (99)	50.6	8531	5.4	164.7
Closure	133 (80)	85.5	18097	638.8	1639.6
Cli	38 (38)	2.5	662	2.4	157.0
Codec	18 (18)	4.5	1051	2.4	87.1
Compress	47 (47)	15.5	4066	6.4	137.6
Total	461 (383)	289.5	45494	763.6	6204.0

1) Subject Programs: We conduct the experimental studies on the real-world benchmark of Defects4J [10]. Table II shows the statistics of subject programs. In total, we considered 383 faults out of 461 faults in Defects4J (v2.0.0). The remaining versions are excluded from the experiments since the omission faults or the faults cannot be detected by the test suites.

2) Evaluation Metrics and Implementation of Experiment: In the experiments, we utilize four metrics (i.e., EXAM [4], Top-N [11], MAP [11], and MTP [4]) to evaluate the effectiveness of our proposed CBTCR. A fault localization technique has a lower EXAM, higher Top-N, and higher MAP demonstrates a better technique. In addition, cost reduction techniques with lower MTP indicate better efficiency [9]. Note that we take the average rank of the statements that shared the same suspiciousness to break the tie [12].

3) Implementation of Experiment: We choose three MBFL test case reduction strategies (FTMES [6], IETCR [7] and the random strategy noted as "RAND") as the baselines. We use 10%, 20%, and 30% to sample the pass test cases followed by the previous studies [3], [4], [7]. We implement Metallaxis [3] (noted as"MBFL") and MCBFL-hybrid-avg [12] (noted as "MCBFL") as MBFL techniques. In addition, we generate mutants by using the mutation tool Major and consider the mutation operators provided by this tool.

B. Results Analysis

TABLE III
Accuracy improvement for MBFL techniques on different sampling ratios

Technique	**Strategy**	**Sampling ratio =**		
		10%	**20%**	**30%**
MBFL	RAND	42.52%	17.71%	33.80%
	IETCR	62.16%	34.52%	19.54%
	FTMES	0.87%	18.36%	24.92%
MCBFL	RAND	22.37%	12.51%	20.33%
	IETCR	28.21%	21.92%	15.38%
	FTMES	26.27%	32.48%	33.92%
Average Improve	RAND	32.44%	15.11%	27.06%
	IETCR	45.18%	28.22%	17.46%
	FTMES	13.57%	25.42%	29.42%

TABLE IV
The *p-value* of CBTCR and other strategies on MBFL techniques at different ratios

Technique	**Strategy**	**Sampling ratio =**		
		10%	**20%**	**30%**
MBFL	RAND	0.0346	0.0037	0.0286
	IETCR	0.0110	0.0040	0.0185
	FTMES	0.0246	0.0018	0.0225
	Original	0.5487	0.9542	0.8124
MCBFL	RAND	0.0444	0.0093	0.0110
	IETCR	0.0464	0.0063	0.0192
	FTMES	0.0668	0.0217	0.0032
	Original	0.8371	0.8220	0.8818

1) ***Answer for RQ1 (The effectiveness of CBTCR):*** Table I shows that, on MBFL, CBTCR locates 122 faults at Top-5 with a sampling ratio of 10% , which more than RAND (103), IETCR (93), and FTMES (120). We can find that

FTMES can locate more faults at Top-1 (43). One possible reason is that some passed tests cannot help in detecting faults. Also, CBTCR performs better when sampling 20% and 30% passed test cases at the metric of MAP. Besides, CBTCR performs similarly to the original MBFL (noted as "Original") when sampling 30% passed test cases. On MCBFL, CBTCR outperforms the other three strategies in all cases. Moreover, Table III shows the accuracy improvement for each strategy with different sampling ratios compared with CBTCR at MAP. The last row shows that CBTCR improves other strategies ranging from 13.57% to 45.18% in fault localization accuracy on average. Besides, CBTCR locates more faults at the metric of Top-3(115) when the sampling ratio equals 10% than Original. In addition, the wilcoxon signed-rank test (at a confidence level of 95% and $p\text{-}values$ less than 0.05 means significant) in Table IV shows that EXAM of MBFL techniques with CBTCR has a significant difference with other three strategies and the original MBFL. Therefore, we regard 10% as the trade-off sampling ratio for the CBTCR strategy.

*2) **Answer for RQ2 (The cost of CBTCR):*** Table V further presents the detailed average reduction ratios MTP of each strategy. From Table V, we can see that CBTCR reduces execution cost varied from 69.74% to 89.47% and it reduces 85.43% of the cost on average. Besides, CBTCR costs more 8.84% than FTMES, but it improves 13.57% of the fault localization accuracy for FTMES on average (see Table III). Therefore, CBTCR is a trade-off strategy that obtains better fault localization effectiveness by losing some efficiency of running this strategy. Hence, there still exists room for boosting the efficiency of CBTCR.

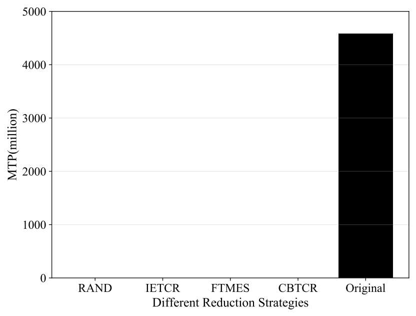

Fig. 2. MTP of CBTCR and other strategies at sampling ratio 10%

TABLE V
AVERAGE MTP EXECUTION RATIOS REDUCED BY DIFFERENT STRATEGIES

Project	**RAND**	**IETCR**	**FTMES**	**CBTCR**
Lang	89.23%	89.23%	99.00%	89.23%
Chart	89.47%	89.47%	99.41%	89.47%
Time	87.64%	89.81%	97.05%	87.64%
Math	86.09%	86.09%	95.13%	86.09%
Closure	69.74%	73.64%	74.67%	69.74%
Cli	88.93%	88.93%	98.68%	88.93%
Codec	87.02%	87.02%	96.17%	87.02%
Compress	85.35%	85.35%	94.09%	85.35%
Average	85.43%	86.19%	94.27%	85.43%

V. Conclusion and Future Work

This paper proposes a novel contribution-based test case reduction (CBTCR) for mutation-based fault localization (MBFL). CBTCR measures the contribution value of test cases using test coverage and SBFL suspiciousness, and then the test cases with higher values are used for running the mutants. CBTCR is a strategy that uses both failed test cases and some valuable passed test cases. This paper evaluates CBTCR by conducting an empirical experiment on 383 faults from Defects4J. The experimental results demonstrate that CBTCR outperforms the previous test case reduction strategies (IETCR and FTMES). Also, the results indicate CBTCR can highly improve the efficiency of MBFL by reducing 85.43% of the cost on average, while maintaining almost the same fault localization accuracy as original MBFL techniques. In the future, we plan to investigate the theory of CBTCR and extend our strategy to more large-scale programs.

Acknowledgment

The work is supported by the National Quality Infrastructure of China (Grant no. 2022YFF0608103).

References

[1] N. Bayati Chaleshtari and S. Parsa, "Smbfl: Slice-based cost reduction of mutation-based fault localization," *Empirical Software Engineering*, vol. 25, pp. 4282–4314, 2020.

[2] W. E. Wong, V. Debroy, R. Gao, and Y. Li, "The dstar method for effective software fault localization," *IEEE Transactions on Reliability*, vol. 63, no. 1, pp. 290–308, 2013.

[3] M. Papadakis and Y. Le Traon, "Metallaxis-fl: mutation-based fault localization," *Softw. Test. Verification Reliab.*, vol. 25, no. 5-7, pp. 605–628, 2015.

[4] Z. Li, H. Wang, and Y. Liu, "Hmer: A hybrid mutation execution reduction approach for mutation-based fault localization," *J. Syst. Softw.*, p. 110661, 2020.

[5] M. Papadakis and Y. Le Traon, "Effective fault localization via mutation analysis: a selective mutation approach," in *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. ACM, 2014, pp. 1293–1300.

[6] A. A. L. de Oliveira, C. G. Camilo-Junior, E. N. de Andrade Freitas, and A. M. R. Vincenzi, "Ftmes: A failed-test-oriented mutant execution strategy for mutation-based fault localization," in *Proceedings of the International Symposium on Software Reliability Engineering*. IEEE, 2018, pp. 155–165.

[7] H. Wang, B. Du, J. He, Y. Liu, and X. Chen, "Ietcr: An information entropy based test case reduction strategy for mutation-based fault localization," *IEEE Access*, vol. 8, pp. 124 297–124 310, 2020.

[8] Y. Liu, Z. Li, R. Zhao, and P. Gong, "An optimal mutation execution strategy for cost reduction of mutation-based fault localization," *Inf. Sci.*, vol. 422, pp. 572–596, 2018.

[9] Y. Liu, Z. Li, L. Wang, Z. Hu, and R. Zhao, "Statement-oriented mutant reduction strategy for mutation based fault localization," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2017, pp. 126–137.

[10] R. Just, D. Jalali, and M. D. Ernst, "Defects4j: A database of existing faults to enable controlled testing studies for java programs," in *Proceedings of the 2014 international symposium on software testing and analysis*, 2014, pp. 437–440.

[11] H. Wang, Z. Li, Y. Liu, X. Chen, D. Paul, Y. Cai, and L. Fan, "Can higher-order mutants improve the performance of mutation-based fault localization?" *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 1157–1173, 2022.

[12] S. Pearson, J. Campos, R. Just, G. Fraser, R. Abreu, M. D. Ernst, D. Pang, and B. Keller, "Evaluating and improving fault localization," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 609–620.