An Empirical Study of Mutation-Based Test Case Clustering Prioritization and Reduction Technique

Longbo Li, Yanhui Zhou, Yong Yu, Feiyan Zhao
Southwest University, Chongqing, China

Shenghua Wu and Zhe Yang
Meiyun Zhi Number Technology Co., Ltd.

Mutation analysis is also called mutation testing, which is a method for measuring the quality of test suite, using mutants (artificially injected faults) generated from the original program using special rules (mutation operators)[1].

[1] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," Computer, vol. 11, no.4, 1978, pp. 34-41.

Due to the expansion of the the software scale, a large number of test cases are generated in regression testing.

↓

We need to prioritize and reduce test cases during the regression testing.

We usually prioritize and reduce test cases usually using code coverage criterion.

1. Statement coverage[1]

2. Branch coverage[2]

A test case can kill the mutation program, not only cover code information, also need to kill the mutation program.

[1]

Mutation criterion > Code Coverage criterion

[1] S. Elbaum, A. Malishevsky, and G. Rothermel, "Prioritizing Test Cases for Regression Testing," IEEE Transactions on Software Engineering,vol.27, no.10, 2001, pp. 924-948.
[2] Yoo S, Harman M. "Regression testing minimization, selection and prioritization: A survey," Software Testing, Verification and Reliability 2012, 22(2):pp. 67-120.

Clustering algorithm belong to unsupervised learning algorithm. Clustering algorithm can reveal the intrinsic properties and laws of data. Hierarchical clustering algorithm(HCA) to achieve good results based on code coverage information in test case prioritization[1].

We use HCA to explore mutation-based test case prioritization and reduction technique.

[1]R. Carlson, H. Do, and A. Denton. “A clustering approach to improving test case prioritization: An industrial case study,” In Proc, ICSM, 2011,pp. 382-391.

1. Every mutation operator can generated many mutants in mutation analysis. If a mutant and original program return different returns for same the test, then the test kills the mutant.
2. The priorities of the mutants are the same in previous study[1][2][3].

Mutants have diverse attributes, and each mutant has a different priority.

[1]Do H, Rothermel G. "On the use of mutation faults in empirical assessments of test case prioritization techniques," IEEE Transactions on Software Engineering 2006, 32(9),pp. 733-752.
[2] Lou Y, Hao D, Zhang L. "Mutation-based test-case prioritization in software evolution," In Proceedings of the 26th International Symposium on Software Reliability Engineering (ISSRE), IEEE: Gaithersbury, MD, USA, 2015,pp. 46-57.
[3] Shin D. Yoo S. Papadakis M. and Bae D.H."Empirical evaluation of mutation-based test case prioritization techniques,"Software:testing verification and reliability, Vol.29.2019.e1695.

2 Mutation Program Unit

Mutation Program Unit(MPU) Definition:

We call mutation programs generated by the same mutation operator a mutation program unit. A MPU contains many mutants. The smallest unit in MPU is a mutant. In this paper, we use the smallest mutation program unit.

2 Mutation Program Unit

Why do we use the MPU?

1. **A mutation program that is hard to kill by a test case can usually trigger a real fault[1].**

2. **Because of the large number of mutation programs generated, mutation analysis consumes a lot of computing resources, and the use of high-priority mutation programs can reduce computational overhead.**

[1] Shin D. Yoo S. Papadakis M. and Bae D.H.“Empirical evaluation of mutation-based test case prioritization techniques,”Software:testing verification and reliability, Vol.29.2019.e1695.

2 Mutation Program Unit Kill & Priority Matrix

Test Case Name	Mutation Program Unit(MPU)							
	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
T_0	1	0	**1**	0	0	1	0	0
T_1	1	1	**1**	0	1	0	0	0
T_2	0	1	**1**	1	1	1	0	0
T_3	0	0	**1**	0	0	0	1	1
T_4	0	0	0	0	0	0	0	0

Test Case Name	Mutation Program Unit(MPU)							
	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
T_0	0.5	0	**0**	0	0	0.5	0	0
T_1	0.5	0.5	**0**	0	0.5	0	0	0
T_2	0	0.5	**0**	0.75	0.5	0.5	0	0
T_3	0	0	**0**	0	0	0	0.75	0.75

After the mutation analysis, we first generate the mutation program unit kill matrix, which is then transformed into the mutation program unit priority matrix.

Average Percentage of Fault Detection(APFD)[1]

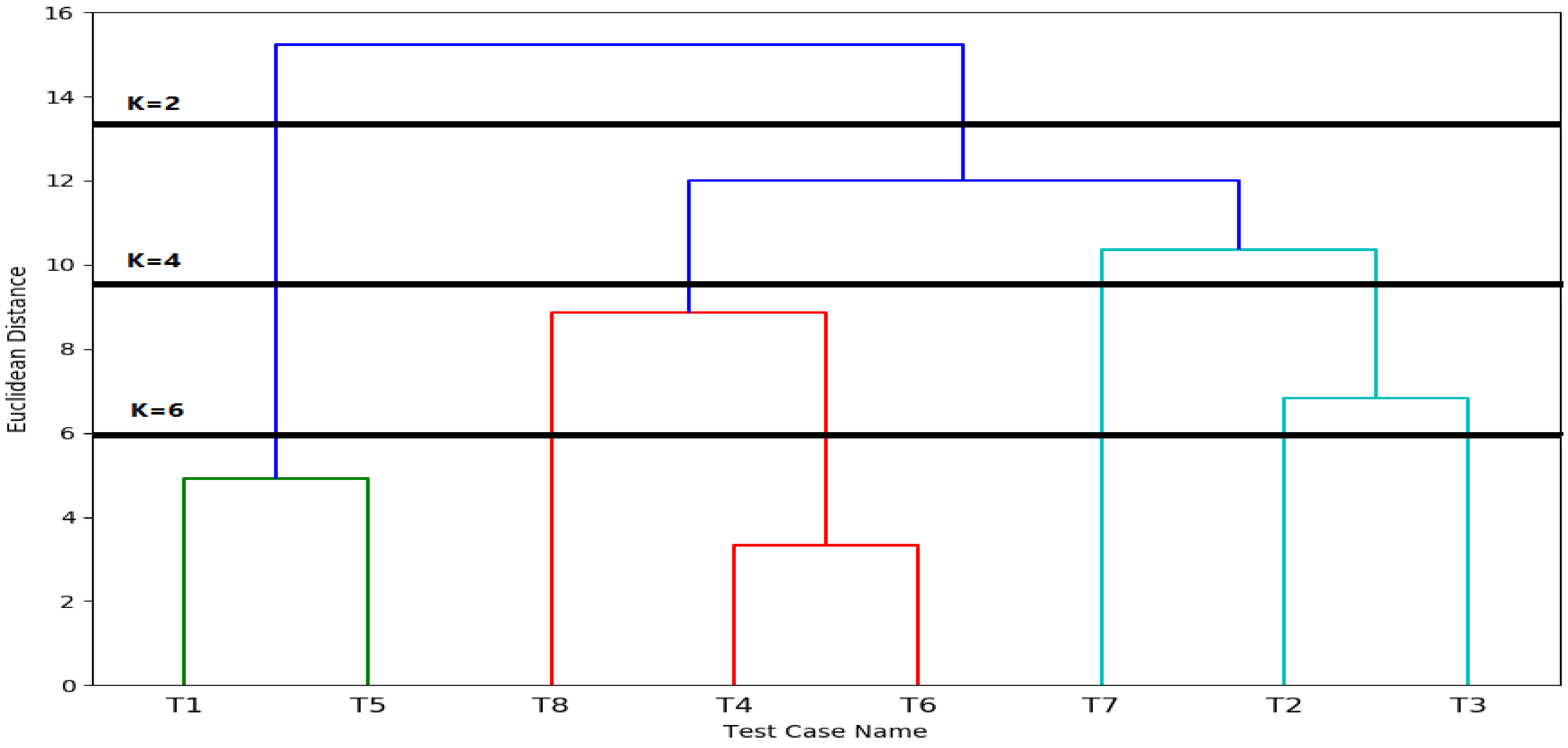
$$APFD = 1 - \frac{TF_1 + \cdots + TF_n}{nm} + \frac{1}{2n}$$

Average Percentage of Weight Fault Detection(APWFD)

$$APWFD = 1 - \frac{W_1 \times TF_1 + \cdots + W_n \times TF_n}{\sum_{j=1}^{n} W_i \times n} + \frac{1}{2n}$$

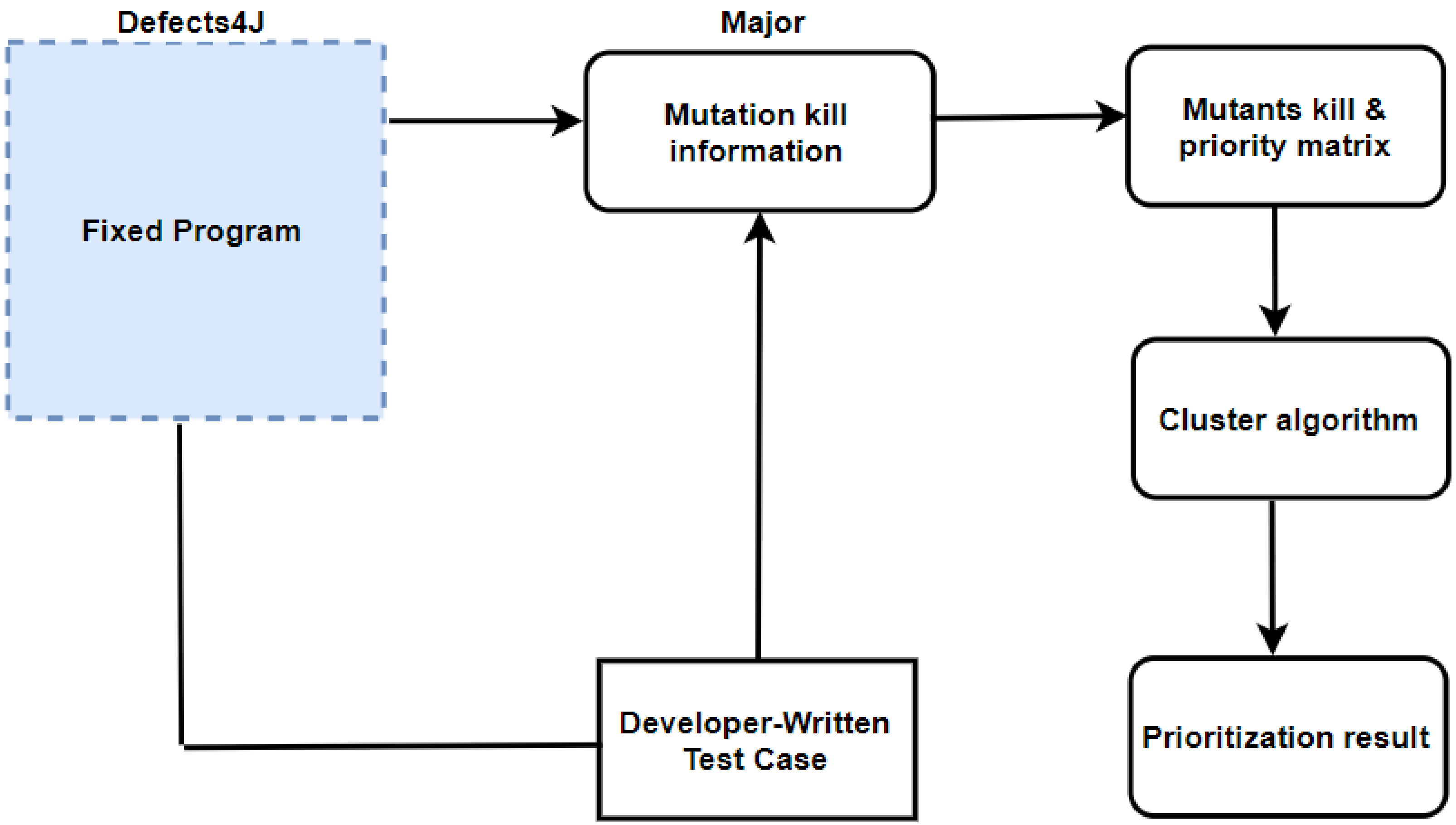
[1] Elbaum S, Malishevsky A, Rothermel G. “Test case prioritization: A family of empirical studies,” IEEE Transactions on Software Engineering 2002; 28(2):159-182.

3 Hierarchical Clustering Algorithm

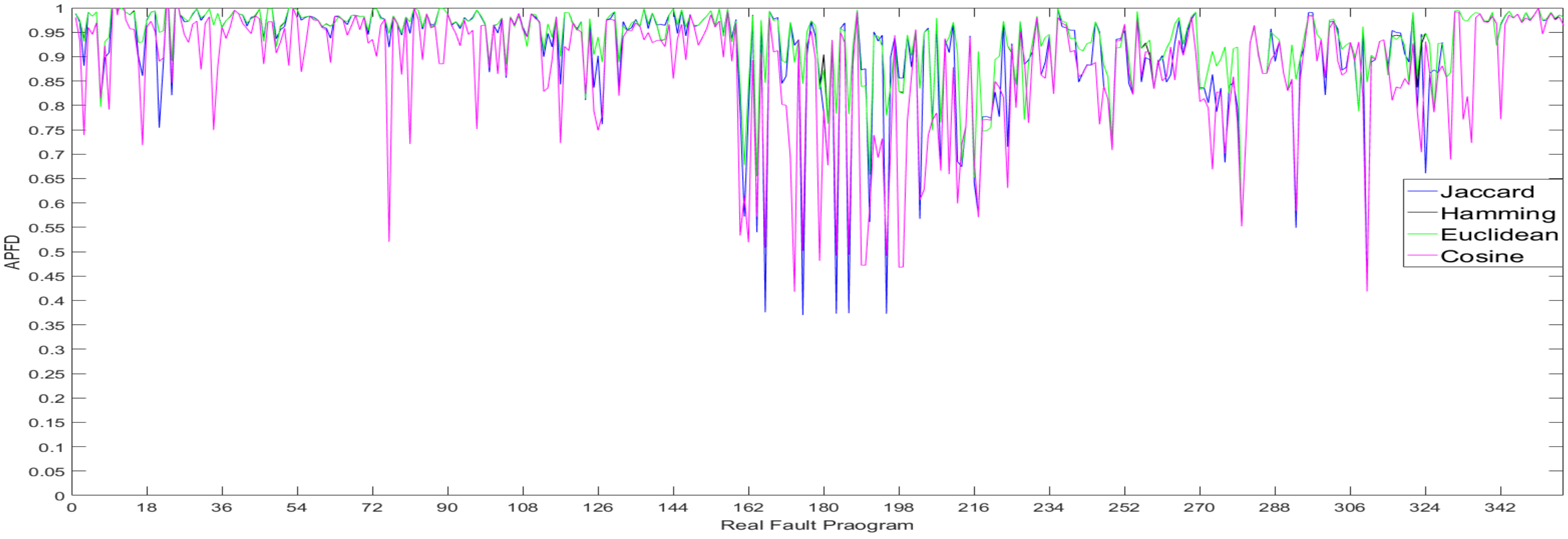


Hierarchical clustering algorithm can control how many clusters are generated. K represents the number of clusters.

3 Experiment Setup



Is the different distance calculation method have different effects on test case prioritization and reduction?



Does our approach have an impact on the capability of fault detection test cases?

Program	Analysis index			Analysis index		
	APFD	Cluster time(s)	Test case	APWFD	Cluster time(s)	Test case
Chart	20.0210	2.0764	5693	18.7334	0.1189	3577
Closure	123.9185	1649.2548	440296	123.1317	758.1702	182155
Lang	47.6511	10.0078	11338	48.4177	1.5700	6480
Math	91.4900	165.7274	22688	88.2791	63.9586	9941
Time	25.5815	122.3370	70239	25.2988	71.4638	17935
Average	0.8819	5.5697	-	0.8681	2.5579	-

CONCLUSION

1. We have proposed an novel definition to Mutation Program Unit.
2. We have proposed mutation-based test case prioritization and reduction method.
3. Our method can reduce the number of test cases by 40%, and the loss of fault detection capability is only 1.38%.

FUTURE WORK

1. We will systematically explain the theoretical mutation program unit priority we propose.
2. We will consider reducing the complexity of algorithms.
3. We will explore the impact of our approach on test cases that trigger real faults.

THANKS

lilongbo@email.swu.edu.cn