A Model to Estimate First-Order Mutation Coverage from Higher-Order Mutation Coverage

Ali Parsai, Alessandro Murgia, and Serge Demeyer
University of Antwerp
Antwerp, 2020
ali.parsai, alessandro.murgia, serge.demeyer@uantwerpen.be

Abstract

The test suite is essential for fault detection during software evolution. First-order mutation coverage is an accurate metric to quantify the quality of the test suite. However, it is computationally expensive. Hence, the adoption of this metric is limited. In our study, we addressed this issue by proposing a realistic model able to estimate first-order mutation coverage using only higher-order mutation coverage. Our study shows how the estimation evolves along with the order of mutation. We validated the model with an empirical study based on 17 open-source projects.

1 Introduction

The advent of agile processes with their emphasis on test-driven development and continuous integration implies that developers want (and need) to test their newly changed or modified classes or components early and often. Therefore, the quality of the test suite is an important factor during the evolution of the software. One of the extensively studied methods to improve the quality of a test suite is mutation testing [DLS78]. Although the idea of mutation testing has been introduced in the late 1970s [DLS78], it has not found widespread use in industry due to its computationally expensive nature. Therefore, several approaches have been proposed in order to make this technique feasible in industrial settings [OU01]. Exploiting higher-order mutants instead of first-order mutants is one way used in literature to approach this problem [JH11]. A first-order mutant is created by injecting a single fault into the source code of a program. Similarly, a higher-order mutant is created by injecting several faults into the source code. Higher-order mutants can be created by partitioning the set of first-order mutants randomly, and combining first-order mutants in each partition into higher-order mutants. The benefits of higher-order mutation are twofold. First, higher-order mutants are less likely to generate false positives than first-order mutants [JH09]. Equivalent mutants act as false positives [GSZ09], and their detection is an undecidable problem. However, when an equivalent mutant is combined with a non-equivalent mutant, the resulting higher-order mutant is non-equivalent as well [KPM12]. This means that higher-order mutants are less likely to suffer from the equivalence problem [JH08]. Second, by using higher-order mutants, fewer mutants need to be evaluated [JH08]. For instance, combining first-order mutants two-by-two into second-order mutants would reduce the number of mutants to 50%. As a consequence, it saves half of the computational time of mutation testing.

Despite the benefits, the adoption of higher-order mutants presents also drawbacks and limitations. When higher-order mutants are constructed in the aforementioned manner, the mutation coverage (percentage of killed mutants to all non-equivalent mutants) calculated using higher-order mutants is not as precise as the one calculated using first-order mutants, with the former often overestimating the capabilities of a test suite. While using higher-order mutants allows us to evaluate less mutants, it also means that we lose information regarding the status of each underlying first-order mutant. This creates a limitation on the order of mutation, as the value of the lost information overcomes the value of the information at hand.

In our published paper, we addressed these shortcomings [PMD16]. We proposed a model able to esti-

Proceedings of the Seminar Series on Advanced Techniques and Tools for Software Evolution SATToSE 2017 (sattose.org).
07-09 June 2017, Madrid, Spain.

mate first-order mutation coverage requiring only the computation of higher-order mutation coverage and yet with a smaller chance of equivalence problem. Initially, we evaluated the estimation of our model for the second-order mutants. Then, we extended the analysis to find out whether our model can be used with mutants of third or higher order. For each higher-order mutant, we also verify whether our model correctly describes their real behavior. More specifically, we look for side effects due to the combination of first-order mutants into higher-order mutants.

The rest of the paper is structured as follows: Section 2 describes our proposed model. In Section 3 we explain the design of our case study. In Section 4 we present the results of our experiment. Finally, we conclude the paper in Section 5.

2 Proposed Model

We define a higher-order mutant as a combination of multiple first-order mutants: $h = m_1, m_2, ..., m_n$. We assume that the higher-order mutant is killed if and only if at least one of the underlying first-order mutants are killed (Expected category).

The probability of a mutant ($P(mutant)$) being killed is defined as the likelihood of choosing a random mutant with a killed status out of all mutants. The probability of a higher-order mutant being killed ($P(h)$) can be calculated from the probability of underlying first-order mutants ($P(m_i)$) in the following manner:

$$\begin{aligned} 1 - P(h) =& (1 - P(m_1)) \times (1 - P(m_2|m_1)) \times ... \\ & \times (1 - P(m_n|m_1, m_2, ..., m_{n-1})) \end{aligned} \quad (1)$$

This simply means that the probability of a higher-order mutant being killed can be evaluated by an ordered evaluation of the probabilities of underlying first-order mutants being killed.

After calculating the higher-order mutation coverage, we assume all higher-order mutants have the same probability of being killed equal to this mutation coverage.

For the sake of estimation, we assume all first-order mutants have the same equal probability of being killed, and this probability is equal to first-order mutation coverage. We assume these probabilities to be independent from each other:
$P(m_1) = P(m_2|m_1) = ... = P(m_n|m_1, m_2, ..., m_{n-1})$

To ease the calculation of mutation coverage, we simplify Equation 1 using the above assumptions:

$$1 - P(h) = (1 - P(m))^n \quad (2)$$

Using Equation 2, we assume $P(h)$ equals to the higher-order mutation coverage, i.e. any selected higher-order mutant from a set containing $n\%$ killed higher-order mutants is a killed one with the probability of $n\%$. From this equation, we derive the value for $P(m)$, which is the probability of a single first-order mutant being killed, and we use this value as our estimation for first-order mutation coverage (Equation 3).

$$P(m) = 1 - \sqrt[n]{1 - P(h)} \quad (3)$$

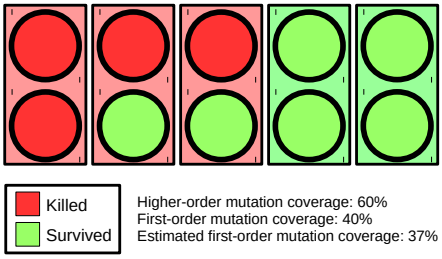

Figure 1: Example of the estimation provided by the model

Figure 1 shows an example of this model; circles denote first-order mutants and rectangles are second-order mutants created by combining two first-order mutants. Out of 10 first-order mutants, 4 are killed, therefore the mutation coverage for this example is equal to 40%. However, 3 out of 5 higher-order mutants are killed, resulting in 60% coverage calculated using higher-order mutants. Considering our model, we estimate the first-order mutation coverage using Equation 3 as 37%.

3 Case Study Design

In order to evaluate our model, we performed first-, second-, third-, fourth-, fifth-, sixth-, and eighth-order mutation testing on 17 different Java projects. For these projects, 20,849 first-order mutants were generated for a total of 1,022 classes.

Cases

We selected 17 open-source projects for our empirical study (Parsai et al. Table II [PMD16]). The selected projects differ in size of their production code, test code, number of commits, and team size to provide a wide range of possible scenarios. Moreover, they also differ in adequacy of the test suite based on statement, branch, and mutation coverage. All selected projects are written in Java.

Model Evaluation

The accuracy of the model is affected by the accuracy of $P(m)$. This in turn depends on the number of mutants generated for the class in question (10 mutants would result in a probability with 0.1 accuracy, while 100 mutants improves this to 0.01). We define a

threshold t as the number of minimum generated mutants for a class, and we filter out the classes with less than t generated mutants. The higher the threshold, the fewer the classes considered. This can be seen in Figure 2, where the x axis is the threshold, and the y axis is the number of remaining classes after applying the threshold. In our experiment, we decided to evaluate how the model behaves with different values of t in order to find a value that still keeps as many classes as possible, while filtering out less accurate data.

The first goal of our work is to find a model that estimates—as best as possible—the first-order mutation coverage using higher-order mutants coverage. The second one is to create a model able to justify the obtained estimation. For the first goal, a simple polynomial regression analysis is used in order to find the best fitting curve of the empirical data. The order of the polynomial function influences the quality of the fitting: the higher the order, the higher the chance that the curve (over)fits the all points. However, to satisfy the second goal, we also need to find a curve that describes the *real* behavior of the empirical data. For this reason, we cannot use a polynomial fitting curve of third or higher order, since a polynomial curve of third order has 2 inflections, namely there exist a range of values where an increment of the first-order coverage leads to a decrement of the second-order coverage. The latter event cannot happen since it would imply that the more first-order mutants are killed, the less higher-order mutants are killed. In our experiment, we compare the best second order fitting curve with the one provide by our model. This allows us to assess how far our model is from the optimal estimation.

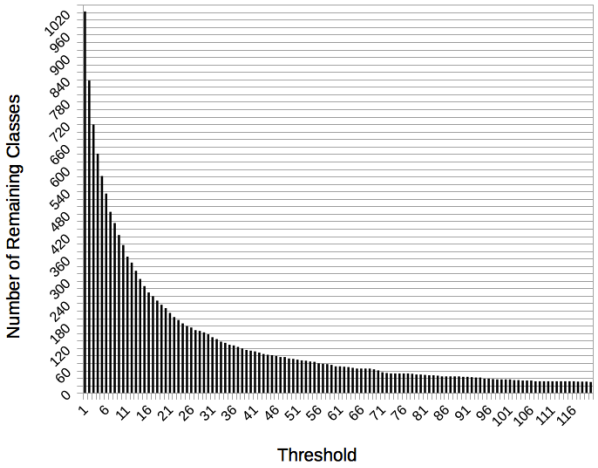

Figure 2: Number of remaining classes after applying threshold t

4 Results and Discussion

In this section, for each research question, we first discuss the motivation, then we explain our approach, and finally we present our findings.

RQ1. To what extent the second-order mutants can be used in place of the first-order mutants for estimation purposes?

We want to verify the applicability of the model, and for this reason, we inspect two aspects:

Estimation Accuracy. How much does the second-order mutation coverage overestimates the capabilities of the test suite.

Soundness of the Model. We need to verify whether our assumptions regarding the mutant interactions hold with the empirical data.

For this reason, we break the first research question in two parts:

a) How accurately can our model estimate the first-order mutation coverage? To evaluate the accuracy of

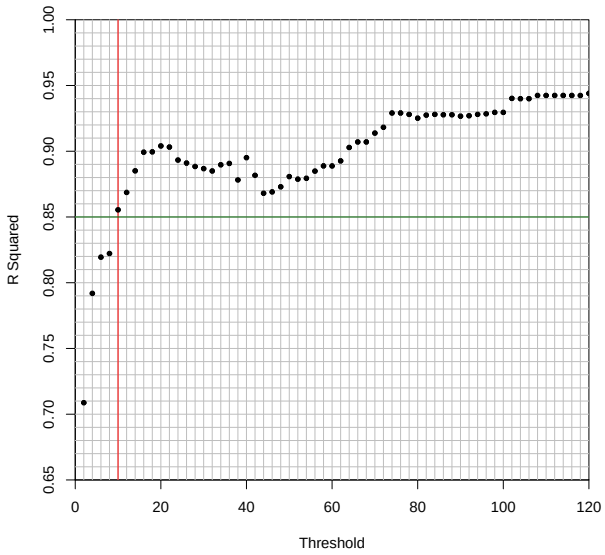

Figure 3: R^2 between estimated results and empirical data for different thresholds for second-order mutation. The red line denotes the chosen threshold, and the green line shows the level of R^2 that the chosen threshold guaranties.

Table 1: Parameters of the estimated curve

Parameter	Estimated Value	Standard Error	p-value
x^0	0.005342	0.010109	0.598
x^1	1.999221	0.037831	$< 2 \times 10^{-16}$
x^2	-1.008797	0.032452	$< 2 \times 10^{-16}$

the estimations, we calculate the coefficient of determination (R^2) between the results estimated by our model, and the empirical data.

Figure 3 shows the value of R^2 between estimated results and empirical data for various values of the threshold t. The estimations achieve an R^2 value 0.85 with a threshold t as low as 10. Using such threshold, the model can still be applicable to 38.7% of the classes which account for 89.8% of the mutants. For higher thresholds, the R^2 value is always higher than 0.85. These results show that the model can provide a good accuracy in estimating the first-order mutation coverage from the second-order mutation coverage. Yet, it halves the computation time required.

Figure 4 shows the empirical data for $t = 10$ and the estimated curve. The blue dots denote a class with an x value of first-order mutation coverage and a y value of second-order mutation coverage. The red curve is the one predicted by our model: $y = -x^2+2x$. The curve determined by the regression analysis is $y = -1.008797x^2 + 1.999221x + 0.005342$. Table 1 shows the p-value for each estimated parameter. Considering the very low p-values calculated for each parameter, we can safely say that the trend of the empirical data is well represented by the best fitting curve. As we can see, the curve predicted by our model, and the one provided by the regression analysis are very close in terms of coefficients. This highlights that our model provides (almost) the best possible estimations. The accuracy of the estimation, however, changes according to the mutation coverage value. This is observed especially for classes with a very high second-order mutation coverage, in which there is less information available for the model.

b) Is the modeled behavior of second-order mutants consistent with the empirical data?

We categorize the second-order mutants into three categories: Expected, HK-FS, and HS-FK. The higher the number of mutants in the Expected category, the better the model describes the behavior of the second-order mutants.

Out of 10,153 second-order mutants generated for all projects, 10,069 (99.17%) are in the Expected category. This means that overall only 84 (0.83%) mutants are of unexpected status, of which 13 (0.13%) mutants belong to the HK-FS category, and 71 (0.70%) mutants belong to the HS-FK category. In total, 764 (91.28%) out of 837 classes do not contain any HS-FK or HK-FS mutants. In almost all projects the number of mutants in the Expected category is higher than 97%. The only exception is given by the project *ScribeJava*. By manually checking this case, we found that the unexpected second-order mutants were created in very small classes where the underlying first-order mutants directly interact with each other. Overall, we see that the modeled behavior applies to the vast majority of the second-order mutants.

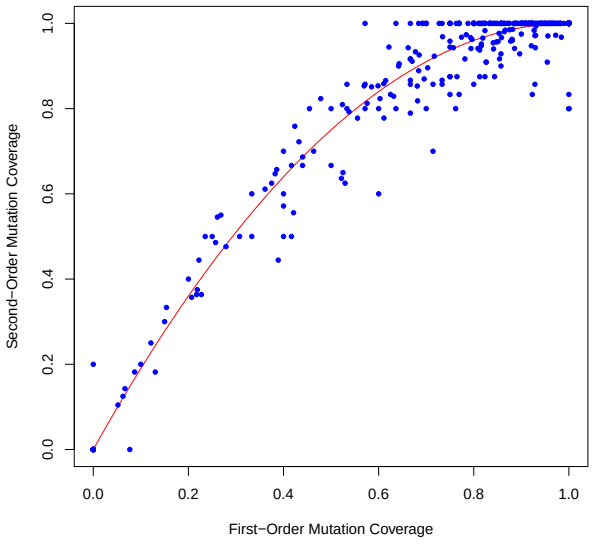

Figure 4: Empirical data and the estimated curve for $t = 10$. The red curve is the estimation provided by the model, and each blue dot represents a class.

RQ2. How does incrementing the order of mutation affect the accuracy of the estimation provided by the model?

For all projects, we perform third-, fourth-, fifth-, sixth-, and eighth-order mutation testing. Then, we observe how the order of mutation affects the accuracy of the estimations for various thresholds. We also verify whether the model describes the real behavior by computing the number of mutants in the Expected category for the aforementioned orders of mutation.

Figure 5 shows the computed R^2 between estimated results and empirical data for various thresholds t for second, third, fourth, fifth, sixth and eighth orders of mutation. In this figure, we observe that with the increment of the order of mutation, the R^2 values decrease for any threshold. This can be attributed to the fact that the higher the order of the mutant is, the higher is the chance that at least one of the underlying first-order mutants is killed. This means that increasing the order of mutation makes the model less accurate. From the figure we notice that the values of

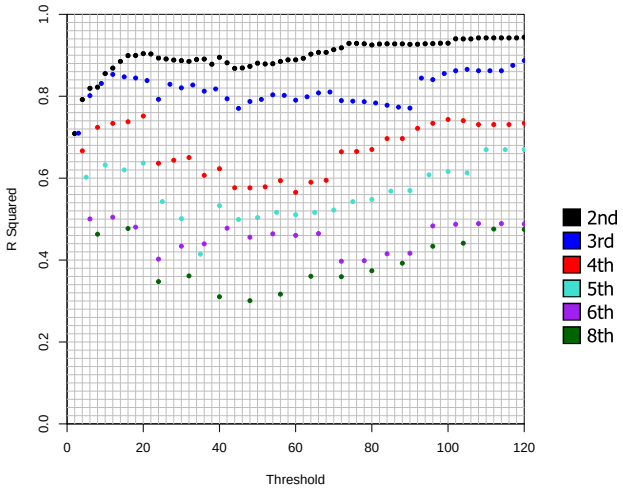

Figure 5: R^2 between estimated results and empirical data for different thresholds.

R^2 remain similar for the second-order and the third-order within $2 \leq t \leq 12$. Whereas, the values of R^2 for fourth and higher orders are visibly lower, highlighting the limitation of proposed model.

5 Conclusion

First-order mutation coverage is an effective metric to evaluate the quality of the test suite. However, its adoption is hindered due to the high computational time consumption. On one hand, the adoption of higher-order mutants to evaluate the quality of the test suite is a viable solution. On the other hand, this approach provides less realistic estimations than the ones obtained with first-order mutants. As a solution, we propose a realistic model able to estimate first-order mutation coverage based on higher-order mutation coverage. The benefits of this model are that (i) we achieve a good accuracy in estimating the first-order mutation coverage based on second- and third-order mutation coverage, (ii) we halve the computational time that would be required by first-order mutation testing (at minimum). Moreover, the chance of existence of equivalent mutants is less than first-order analysis, even though the model does not explicitly consider them in its estimation. Finally, the model correctly describes the real behavior of vast majority of higher-order mutants.

References

[DLS78] Richard A. DeMillo, Richard J. Lipton, and F. G. Sayward. Hints on test data selection: Help for the practicing programmer. *Computer*, 11(4):34–41, apr 1978.

[GSZ09] Bernhard J. M. Grün, David Schuler, and Andreas Zeller. The impact of equivalent mutants. In *Proc. ICSTW 2009 (International Conference on Software Testing, Verification, and Validation Workshops, 2009)*, pages 192–199. Institute of Electrical & Electronics Engineers (IEEE), April 2009.

[JH08] Yue Jia and Mark Harman. Constructing subtle faults using higher order mutation testing. In *Proc. SCAM 2008 (Eighth IEEE International Working Conference on Source Code Analysis and Manipulation)*, pages 249–258. Institute of Electrical & Electronics Engineers (IEEE), sep 2008.

[JH09] Yue Jia and Mark Harman. Higher order mutation testing. *Information and Software Technology*, 51(10):1379–1393, 2009. Source Code Analysis and Manipulation, SCAM 2008.

[JH11] Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, sep 2011.

[KPM12] Marinos Kintis, Mike Papadakis, and Nicos Malevris. Isolating first order equivalent mutants via second order mutation. In *Proc. ICST 2012 (Proceedings of the 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation)*, pages 701–710. Institute of Electrical & Electronics Engineers (IEEE), apr 2012.

[OU01] A. Jefferson Offutt and Roland H. Untch. Mutation 2000: Uniting the orthogonal. In W.Eric Wong, editor, *Mutation Testing for the New Century*, volume 24 of *The Springer International Series on Advances in Database Systems*, pages 34–44. Springer US, 2001.

[PMD16] Ali Parsai, Alessandro Murgia, and Serge Demeyer. A model to estimate first-order mutation coverage from higher-order mutation coverage. In *Proc. QRS 2016 (IEEE International Conference on Software Quality, Reliability and Security)*, pages 365–373. Institute of Electrical and Electronics Engineers (IEEE), Aug 2016.