On Sufficiency of Mutants

Akbar Siami Namin and James H. Andrews
Department of Computer Science, The University of Western Ontario, London, Ontario, Canada
{asiamina, andrews}@csd.uwo.ca

Abstract

Mutation is the practice of automatically generating possibly faulty variants of a program, for the purpose of assessing the adequacy of a test suite or comparing testing techniques. The cost of mutation often makes its application infeasible. The cost of mutation is usually assessed in terms of the number of mutants, and consequently the number of "mutation operators" that produce them. We address this problem by finding a smaller subset of mutation operators, called "sufficient", that can model the behaviour of the full set. To do this, we provide an experimental procedure and adapt statistical techniques proposed for variable reduction, model selection and nonlinear regression. Our preliminary results reveal interesting information about mutation operators.

1. Introduction

We can assess the adequacy of a set of test cases through *mutation testing*, the practice of automatically generating possibly faulty versions of the software under test and counting how many are detected. A more recent application of mutation (e.g., [2]) is assessing the effectiveness of software testing techniques, referred to as *mutation analysis*. Recent research [1] suggests that generated mutants can indeed be considered representatives of real faults.

Mutants are generated by applying *mutation operators*. Many mutation operators have been proposed. The diversity of mutation operators leads to the generation of an enormous number of mutants, making the application of mutation infeasibly expensive. For instance, the application of one mutant generator tool to a program with 137 lines of code yields 4937 mutants [5]. Offutt et al. [4] defined a subset of mutation operators as a *sufficient* set if 1) it generates fewer mutants, and 2) its behavior resembles the behavior of all operators. They compared several subsets of a set of 22 mutation operators, to find which subset was the best candidate for a sufficient set.

We revisit the sufficient set problem. Our approach is different from [4] in that we consider a larger number of mutation operators and the whole space of possible subsets of them, and we find the sufficient set of operators by setting up an empirical study and adapting statistical techniques. *Variable reduction*, *model selection*, and *non-linear regression* are the statistical techniques employed, each tackling the problem from a different perspective.

2. Experimental Procedure

It is very unlikely that we will find one unique sufficient subset of mutation operators. In addition to the unique charactericties of each program, the manner of construction of the test suites might affect the mutants' behaviors. For the former issue, we study the seven Siemens [3] subject programs (average size 327 lines of C code), which are supplemented by a relatively large number of test cases. We generate mutants with the Proteum tool, which uses a comprehensive set of 108 mutation operators.

For the latter issue, both random and coverage-based test suites are generated. We study coverage-based test suites with respect to block (BLK), decision (DEC), computation use ($CUSE$) and predicate use ($PUSE$) criteria, as measured by the ATAC coverage tool. We construct three different types of test suite groups: 1) A group $SING$ of test suites having only one test case each; 2) Groups $RAND_i$ of test suites, for $i \in \{10, 20, 50, 100\}$, each of which contains randomly-selected test suites of size i; and 3) Groups BLK, DEC, $CUSE$, and $PUSE$, in which each test suite achieves a certain percentage c of the relevant coverage measure, for $50 \leq c \leq 99$.

We generate 100 test suites in each group for each program. For each test suite, we then compute: 1) Am_i, the *detection ratio* of the mutants generated by the operator μ_i s.t. $1 \leq i \leq 108$, 2) AM, the detection ratio of the mutants generated by all operators. The detection ratios are fractions with the total number of non-equivalent mutants killed by the test suite as divisor and the total number of mutants as dividend. In terms of statistics, the values of Am_i and AM for the test suites in a given group represent the set of predictors and response variables respectively.

3. Statistical Analysis

Our goal is to find a subset of operators μ_i for $1 \leq i \leq 108$ such that by computing the corresponding Am_i for each test suite and fitting a regression line on the detection ratios of the selected operators, we predict AM accurately for all test suites in the group. To address this refined problem, we adapt statistical techniques grouped into three categories:

Model Selection Algorithms: These algorithms aim to construct and search all possible best-fitted linear models that attempt to predict AM more accurately. Given a set of possible predictors, a model selection algorithm selects a variable Am_i having the highest absolute correlation with AM and fits a linear regression between them. The algorithm then proceeds similarly on the residual of the model. We adapt two model selection algorithms: *All-Subset Regression*, a greedy algorithm, by which we construct all possible good models consisting of a desired number of variables; and *Least-Angle Regression*, a less greedy and more accurate algorithm, overcoming the infeasibility traditional forward selection algorithms.

Variable Reduction Techniques: We use these techniques to reduce the number of predictor variables to a smaller set by avoiding less important variables while not losing too much information. We adapt two variable reduction techniques. In both techniques, we repeatedly identify a pair of highly similar variables and eliminate the one that generates more mutants, stopping when all pairs fall below a certain correlation threshold (i.e., .90). In *Elimination-Based Correlation Analysis (EBC)*, we evaluate similarity according to the correlation itself. In *Cluster Analysis (CA)*, we use a proximity matrix to measure the similarities of variables, and form a *dendrogram* to show the similarities. These techniques may retain variables with distinctive profiles that would be discarded by focusing solely on best-fitted models.

Nonlinear Regression Approach: A possible criticism of the above approaches is that they avoid any possible nonlinear relationships that might exist among the set of predictors and response variables. Transforming the data is the main technique in revealing a nonlinear relation. By nonlinear regression, we construct a complementary set of nonlinear models in addition to the linear ones.

4. Preliminary Results

So far the procedure and analyses have been performed only on one program (`tcas`), for the groups $SING$ and $RAND_i$, and using only some of the analyses [5]. Proteum originally generated 4937 mutants from the program. The sufficient set from the EBC analysis predicts AM with only 72 mutants and 7 operators, while offering fair accuracy.

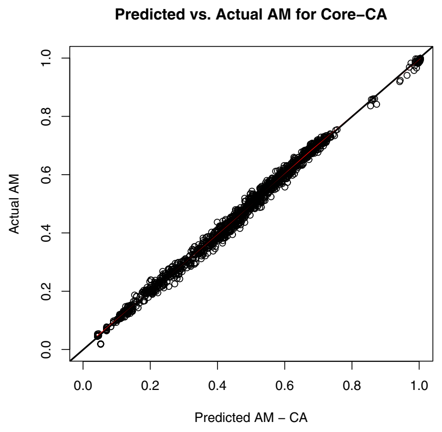

Figure 1. Predicted vs. actual AM

All-subset regression predicts AM with 1393 mutants and 10 operators while offering high accuracy. CA, however, provides a good middle ground, using 366 mutants and 13 operators with relatively good prediction of AM. Figure 1 shows the plot of predicted vs. actual AM for the linear regression model from the sufficient set from CA, for all test suites.

We would like to extend the experiment to larger subject programs and more mutation operators, but the compute-intensive nature of the experiment and analysis (thousands of hours of compute time so far) would make this a challenge. For now, we are extending the experiment and analysis to the other six Siemens programs and the other test suite groups.

References

[1] J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, St. Louis, Missouri, May 2005.

[2] J. H. Andrews, L. C. Briand, Y. Labiche, and A. Siami Namin. Using mutation analysis for assessing and comparing testing coverage criteria. *IEEE Transcations on Software Engineering*, 32(8):608 – 624, August 2006.

[3] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments of the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proceedings of the 16th International Conference on Software Engineering*, pages 191–200, Sorrento, Italy, May 1994.

[4] A. J. Offutt, A. Lee, G. Rothermel, R. H. Untch, and C. Zapf. An experimental determination of sufficient mutation operators. *ACM Transactions on Software Engineering and Methodology*, 5(2):99–118, April 1996.

[5] A. Siami Namin and J. Andrews. Finding sufficient mutation operators via variable reduction. In *Mutation 2006*, Raleigh, NC, USA, November 2006.