Assessing the influence of multiple test case selection on mutation experiments

Assessing the Influence of Multiple Test Case Selection on Mutation Experiments

Marcio E. Delamaro* and Jeff Offutt†
*Computer Systems Department
Universidade de São Paulo, São Carlos, SP, Brazil
Email: delamaro@icmc.usp.br
†Software Engineering
George Mason University, Fairfax, VA, USA
Email: offutt@gmu.edu

***Abstract*—Mutation testing is widely used in experiments. Some papers experiment with mutation directly, while others use it to introduce faults to measure the effectiveness of tests created by other methods. There is some random variation in the mutation score depending on the specific test values used. When generating tests to use in experiments, a common, although not universal practice, is to generate multiple sets of tests to satisfy the same criterion or according to the same procedure, and then to compute their average performance. Averaging over multiple test sets is thought to reduce the variation in the mutation score. This practice is extremely expensive when tests are generated by hand (as is common) and as the number of programs increase (a current positive trend in software engineering experimentation).**

The research reported in this short paper asks a simple and direct question: do we need to generate multiple sets of test cases? That is, how do different test sets influence the cost and effectiveness results? In a controlled experiment, we generated 10 different test cases to be adequate for the Statement Deletion (SSDL) mutation operator for 39 small programs and functions, and then evaluated how they differ in terms of cost and effectiveness. We found that averaging over multiple programs was effective in reducing the variance in the mutation scores introduced by specific tests.

***Index Terms*—Software testing; Mutation testing; Test set selection**

I. Introduction

When conducting experimental studies with mutation testing, researchers usually need to solve some problems that are seldom automated. This includes identifying equivalent mutants and generating adequate test sets as part of experimental setup.

In a recent paper [1] the authors faced the problem of selecting adequate test sets to evaluate the cost and effectiveness of individual mutation operators. We used the approach pioneered by Frankl [2] of first generating a "universe" of test cases that were adequate for the complete set of mutants, and then selected subsets that were adequate for the specific mutants being studied. The cost and effectiveness of each operator can be measured by assessing some characteristics of the adequate test sets, such as their sizes and their mutation scores when run against the complete set of mutants.

Initially, only one adequate test set was used for each mutation operator. This procedure was criticized because for a given operator Op there can be many different Op-adequate subsets of the universe test set, and thus the results could vary depending on which specific tests are selected. Although intuitively appealing, to the best of our knowledge this concern has never been experimentally verified. So, the question addressed in this paper is whether, and to what extent, the selection of different adequate test sets can influence the experimental results.

The results of this paper indicate two interesting findings. The first is that different test sets can give different results on individual programs. The second is somewhat contradictory, when averaged over the set of 39 programs that we studied, statistically this difference all but disappears. These facts can be useful to future researchers in this area. The second finding indicates that when researchers are interested in averages over programs, one test set is enough. The first, however, brings up a practical question: if the mutation score for the complete set of mutants is sensitive to the specific test cases generated for a reduced set of mutants, will this impact the effectiveness of such test sets against "real world" faults? Moreover, is there a way to identify and select the best (most effective) adequate test sets from among a set of tests that all satisfy the same criterion?

The next section of this short paper presents the experimental setup. Section III presents the results obtained, Section IV presents threats to validity, and Section V presents conclusions and recommendations.

II. Experimental Setup

This section describes the subject programs, the tool used, and the experimental procedure.

a) Subject Programs: Thirty nine programs written in C of varying sizes and from different domains were used as experimental subjects. These programs were extracted from the Siemens program suite [3], text books [4], and the software testing literature. The subject programs varied in size from one to 20 functions, and from seven to 390 lines of code, totaling 189 functions and 2853 lines of code. When needed, we refer to an individual subject as P_i.

Table I summarizes the subject programs. For each, the table shows the number of functions, the number of lines of code, the number of mutants generated by all operators, the number of equivalent mutants, and the number of tests in the mutation-adequate test set.

b) Supporting Tool: This study used the C language mutation tool **Pro**gram **Te**sting **U**sing **M**utants (Proteum) [5] to generate a comprehensive set of faulty programs that were used to evaluate the effectiveness of the test sets. This paper uses the mutation score as a proxy for effectiveness. The set of mutation operators implemented into Proteum follows, as closely as possible, the original C operators defined by Agrawal et al. [6]. The set of all mutants for program P_i is referred as M_i.

c) Adequate Test Sets: For each subject program, an adequate test set was constructed to kill all the mutants generated by Proteum. We refer to the test sets for program P_i as T_i. The size of each T_i is shown in Table I.

Most of the tests were generated by hand by the first author. Some test cases were provided with the Siemens programs[1] and we generated additional tests by hand to kill all non-equivalent mutants. For the remaining programs, we also generated tests by hand to kill all non-equivalent mutants. Equivalent mutants were marked by hand analysis.

After all mutants were killed or marked as equivalent, test cases that did not contribute to killing at least one mutant were removed.

d) Data Collection: For each P_i we selected a reduced set of mutants R_i. These mutants were generated by a single mutation operator, Statement Deletion (SSDL). We chose SSDL for two reasons: (1) it is universally applicable to all programs; and (2) it has been studied in recent papers as an alternative to using the complete set of mutation operators [1, 7, 8].

For each R_i we built 10 test sets that were adequate for SSDL, $TR_{i,1}$ through $TR_{i,10}$. Each $TR_{i,j}$ was built by selecting test cases from the universe set, T_i, until an adequate test set was obtained. Starting with an empty set, a test case was considered and added if and only if it killed at least one additional mutant. For the first set, $TR_{i,1}$, tests were considered in the same order as they appeared in T_i. For the remaining $TR_{i,2}$ through $TR_{i,10}$, tests were considered in random order, starting with a new random seed for that test set.

For each $TR_{i,j}$ two metrics were collected: its size and its mutation score when executed against the complete set of mutants M_i. Then, we collected statistics of distribution and central tendency for each $TR_{i,j}$, including mean, median, minimum and maximum values, and the standard deviation (SD). These numbers are analyzed in the next section.

e) Research Questions: We have two primary questions in this experiment:

RQ1: If different SSDL-adequate test sets are selected from the universe of test cases, how different would they be in terms of effectiveness, as measured by the mutation score against the complete set of mutants?

RQ2: if different SSDL-adequate test sets are selected from the universe of test cases, how different would they be in terms of cost, as measured by their sizes?

[1]Programs `print_tokens`, `print_tokens2`, `replace`, `schedule`, `schedule2`, `tcas` and `totInfo`.

III. Results

Table II shows the effectiveness results for the 39 programs. Each row corresponds to program P_i, and the columns show statistics over the 10 SSDL-adequate test sets. For program P_1 (`boundedQueue`), the lowest mutation score across the 10 test sets was 0.9207 and the highest score was 0.9902. The average (mean) mutation score of all 10 test sets is 0.9600, with a standard deviation of .0216.

The differences between the lowest and highest mutation scores of a given program vary a lot among the programs. For example, the high and low scores for P_4 and P_{19} differ by more than 20%, whereas P_{28} and P_{31} differ by less than 1%. So, if researchers rely on results for individual programs, using only one adequate test set might skew the result. For example, if P_4's lowest scoring test set was used, SSDL would be considered to have only .4752 effectiveness. However, if P_4's highest scoring test set was used, its effectiveness would be found to be 0.7723.

Since several of the smallest programs had particularly high SD, we calculated the Spearman rank correlation [9] between the LOC and SD and between the LOC and Max-Min difference. Two series of numbers are perfectly correlated if the value is 1 or -1, and not at all correlated if the value is 0. The LOC and SD were (negatively) correlated with a value of -.65, and the LOC and Max-Min difference with a value of -.63. Thus, we conclude there is a strong correlation between size and spread, that is, the differences in effectiveness among the test sets are smaller with bigger programs. This is significant because as the programs get bigger, we do not need to use as many test sets in our experiment. This is good news for experimentalists, because creating 10 test sets for a 10 line program is one thing, but creating 10 test sets for a 1000 line program is quite another!

The last row of Table II shows the averages over the 39 programs. The average Min is .9093 and the average Max is .9338, so the SD is very small, only .0071. Running a one-way ANOVA between the 10 sequences of test sets for the 39 subjects we can see that there were no statistically significant differences between mutation score means ($F(9, 380) = 0.312; p = 0.971$).

The results for the number of test cases in each adequate test set are shown in Table III. Since each program has different universes of adequate test cases, with different sizes, we analyze the ratio between the largest and the smallest adequate sets for each program, rather than the difference.

The ratio ranges from a low of 1 (on programs P_4, P_9, P_{15}, and P_{38}) all the way to a high of 7 on P_{17}. Not surprisingly, larger programs usually need more tests. This time, larger programs tended to have a higher SD over the 10 sets of tests, but Spearman's correlation between LOC and SD was only .56. The correlation is positive, indicating that

TABLE I
SUBJECT PROGRAMS.

Program	Prog. Name	Functions	LOC	Mutants	Equiv.	Test Cases
P1	boundedQueue	6	49	1121	99	13
P2	cal	1	18	891	71	8
P3	Calculation	7	46	1118	107	13
P4	checkIt	1	9	104	3	9
P5	CheckPalindrome	1	10	166	20	8
P6	countPositive	1	9	151	9	5
P7	date-plus	3	132	2421	160	44
P8	DigitReverser	1	17	496	43	5
P9	findLast	1	10	198	17	6
P10	findVal	1	7	190	18	7
P11	Gaussian	6	23	1086	19	21
P12	Heap	7	41	1079	98	8
P13	InversePermutation	1	15	576	61	12
P14	jday-jdate	2	49	2821	81	27
P15	lastZero	1	9	173	9	5
P16	LRS	5	51	1132	258	8
P17	MergeSort	3	32	991	48	18
P18	numZero	1	10	151	17	5
P19	oddOrPos	1	9	361	71	7
P20	pcal	8	204	6419	779	49
P21	power	1	11	268	12	9
P22	print_tokens	17	349	4322	542	34
P23	print_tokens2	18	275	4734	664	27
P24	printPrimes	2	35	715	64	7
P25	Queue	6	64	469	25	12
P26	quicksort	1	23	1026	82	13
P27	RecursiveSort	1	17	555	45	8
P28	replace	20	390	$11,100$	2062	142
P29	schedule	18	213	2108	221	45
P30	schedule2	16	195	2626	411	41
P31	Stack	6	56	460	49	11
P32	stats	1	19	884	101	7
P33	sum	1	7	165	11	6
P34	tcas	8	63	2384	428	62
P35	testPad	1	24	629	57	14
P36	totInfo	7	214	6693	678	49
P37	trashAndTakeOut	2	19	599	26	12
P38	twoPred	1	10	246	24	10
P39	UnixCal	4	119	4852	339	27
	Total	**189**	**2853**	**66480**	**7829**	**814**
	Min	**1**	**7**	**104**	**3**	**5**
	Max	**20**	**390**	**11100**	**2062**	**142**
	Average	**4.85**	**73.15**	**1704.62**	**200.74**	**20.87**

the spread grows as the number of tests grows. Again, this is not surprising and is possibly purely a function of size.

The ratio between the largest and smallest test set sizes, however, only has a Spearman's correlation value of -.28 with LOC. Thus, we conclude that differences in test set size is not significantly correlated with program size.

Considering only the averages for the 39 programs, as we did with the mutation score, the smallest mean is 4.79 and the largest is 6.00. Running a one-way ANOVA, we can see that there were no statistically significant differences between test set sizes means ($F(9, 380) = 0.222; p = 0.991$).

As with the mutation scores, there is very little difference between the minimum and maximum number of test cases when averaged over all 39 programs.

IV. Threats to Validity & Limitations

A usual threat in this kind of experimental work is the fact that, no matter which programs we use in the experiment, the results can never be generalized to all programs. To deal with this problem we tried to select a large number of programs from different sources and in different domains. Also in this sense, the small sizes of most programs may represent a threat, but analyzing a large number of large programs is practically impossible

The creation of the universe of test cases, adequate to all the C mutants of each programs was done manually. Using different tests or a different tester could result in different results. However the construction of such sets is extremely time consuming, so it would be impractical to create more than one test set. It is the same for the identification of equivalent mutants. It was done by the first author and, besides his experience with mutation testing and the C language, no other measure was taken to guarantee the quality of such process.

Finally, we used SSDL as the operator for test case selection. Other operators, and indeed, any other test criterion could yield different results. The important issue here is not the actual values obtained, but the analysis of the methodological procedures taken. Thus we see no reason why using other

TABLE II
EFFECTIVENESS OF SSDL

Prog.	Min	Median	Mean	Max	St. Dev.	Max-Min
P1	0.9207	0.9658	0.9600	0.9902	0.0216	0.0695
P2	0.8622	0.9268	0.9230	0.9720	0.0359	0.1098
P3	0.8577	0.9289	0.9214	0.9694	0.0342	0.1117
P4	0.4752	0.6287	0.6267	0.7723	0.0904	0.2971
P5	0.9178	0.9315	0.9342	0.9658	0.0177	0.0480
P6	0.8169	0.9085	0.8930	0.9718	0.0595	0.1549
P7	0.9425	0.9549	0.9577	0.9748	0.0113	0.0323
P8	0.8543	0.9879	0.9744	0.9956	0.0424	0.1413
P9	0.7005	0.7914	0.7807	0.8075	0.0322	0.1070
P10	0.8218	0.8822	0.8736	0.9138	0.0359	0.0920
P11	0.9544	0.9679	0.9663	0.9786	0.0085	0.0242
P12	0.9492	0.9787	0.9757	0.9868	0.0113	0.0376
P13	0.8721	0.9215	0.9157	0.9380	0.0202	0.0659
P14	0.9223	0.9639	0.9583	0.9686	0.0144	0.0463
P15	0.8537	0.8628	0.8689	0.8902	0.0153	0.0365
P16	0.9564	0.9681	0.9707	0.9832	0.0102	0.0268
P17	0.9212	0.9475	0.9483	0.9758	0.0159	0.0546
P18	0.8433	0.9328	0.9306	0.9776	0.0504	0.1343
P19	0.7138	0.7724	0.8110	0.9586	0.0901	0.2448
P20	0.9240	0.9348	0.9421	0.9667	0.0164	0.0427
P21	0.9492	0.9609	0.9609	0.9766	0.0090	0.0274
P22	0.9799	0.9873	0.9871	0.9947	0.0042	0.0148
P23	0.9637	0.9821	0.9807	0.9929	0.0094	0.0292
P24	0.9662	0.9816	0.9808	0.9908	0.0089	0.0246
P25	0.9798	0.9966	0.9944	1.0000	0.0070	0.0202
P26	0.9325	0.9657	0.9593	0.9852	0.0181	0.0527
P27	0.9235	0.9480	0.9431	0.9686	0.0150	0.0451
P28	0.9649	0.9717	0.9707	0.9761	0.0034	0.0112
P29	0.9502	0.9637	0.9651	0.9873	0.0093	0.0371
P30	0.9436	0.9626	0.9588	0.9689	0.0089	0.0253
P31	0.9830	0.9903	0.9888	0.9927	0.0029	0.0097
P32	0.9170	0.9374	0.9458	0.9770	0.0200	0.0600
P33	0.7792	0.8506	0.8526	0.9221	0.0641	0.1429
P34	0.8338	0.8701	0.8674	0.8978	0.0239	0.0640
P35	0.9021	0.9213	0.9247	0.9633	0.0231	0.0612
P36	0.9428	0.9535	0.9557	0.9677	0.0076	0.0249
P37	0.8778	0.8901	0.8988	0.9372	0.0198	0.0594
P38	0.6712	0.7793	0.7725	0.8604	0.0707	0.1892
P39	0.9608	0.9653	0.9655	0.9738	0.0038	0.0130
Avg.	**0.9093**	**0.9232**	**0.9232**	**0.9338**	**0.0071**	**0.0245**

TABLE III
SIZES OF SSDL-ADEQUATE TEST SETS

Prog.	Min	Median	Mean	Max	St. Dev.	Max/Min
P1	3	5.0	5.0	6	0.9280	2.00
P2	3	3.0	3.5	5	0.7265	1.67
P3	5	6.0	6.3	8	0.8819	1.60
P4	2	2.0	2.0	2	0.0000	1.00
P5	3	3.0	3.1	4	0.0000	1.33
P6	1	1.5	1.6	3	0.5270	3.00
P7	13	15.0	15.2	17	1.4142	1.31
P8	1	1.0	1.3	2	0.4410	2.00
P9	2	2.0	2.0	2	0.0000	1.00
P10	1	2.0	1.7	2	0.5000	2.00
P11	3	4.0	4.3	5	0.6667	1.67
P12	2	3.0	3.4	5	1.0138	2.50
P13	2	3.0	3.3	5	0.6009	2.50
P14	4	4.0	4.3	5	0.4410	1.25
P15	1	1.0	1.0	1	0.0000	1.00
P16	2	3.0	3.1	4	0.6009	2.00
P17	1	3.0	3.3	7	1.8028	7.00
P18	1	1.0	1.3	2	0.4410	2.00
P19	1	2.0	1.9	3	0.8333	3.00
P20	14	15.0	15.2	17	1.1180	1.21
P21	2	2.0	2.5	4	0.7265	2.00
P22	7	10.0	10.1	14	1.8708	2.00
P23	4	6.5	6.6	9	1.8105	2.25
P24	1	2.0	2.1	3	0.8660	3.00
P25	6	7.5	7.6	10	0.7071	1.67
P26	1	2.0	2.2	5	0.7817	5.00
P27	1	1.0	1.4	3	0.4410	3.00
P28	16	18.5	19.4	24	2.5495	1.50
P29	10	10.0	10.5	12	0.7265	1.20
P30	7	11.0	10.9	15	1.7401	2.14
P31	4	5.0	5.1	6	0.7071	1.50
P32	1	2.0	1.8	2	0.4410	2.00
P33	1	1.0	1.5	3	0.5000	3.00
P34	10	11.5	11.5	13	0.7071	1.30
P35	3	4.0	3.7	4	0.5000	1.33
P36	8	10.0	10.1	12	1.5366	1.50
P37	3	3.5	3.6	5	0.7071	1.67
P38	2	2.0	2.0	2	0.0000	1.00
P39	7	7.0	7.2	8	0.4410	1.14
Avg.	4.79	5.1	5.2	6	0.2139	1.25

mutation operators would yield different results, although we certainly encourage replication of this study.

V. CONCLUSIONS AND RECOMMENDATIONS

This paper evaluates the effect of using multiple test cases in experimental research. Previous researchers have assumed that selecting only one adequate test set could interfere in the results of cost and effectiveness for mutation operators, and thus created multiple test sets. However, this assumption was made without evidence.

Our results show that there can be significant differences for individual subject programs among different test sets chosen for the same adequacy criterion. These differences were observed for both effectiveness (mutation score) and cost (number of tests). This result makes a case for choosing multiple test sets during experimentation.

However, we found that the differences in effectiveness among different test sets was less with larger programs. Perhaps more importantly, the differences tended to disappear when effectiveness is averaged over a collection of programs (39 in our study). This is not surprising since the difference in the results of the test sets for each program is probably due to the chances of selecting specific test case values. If we had insignificant differences among the 10 test set for any single subject, then that would indicate the practice of using multiple sets is not necessary. The fact that a large number of subjects reduces the individual errors is not particularly related to mutation. It is only a good experimental practice.

Thus, we recommend using multiple test sets if only a few subjects are chosen, but if many subjects are used, using multiple test sets may not increase the accuracy of the experimental work. It is important to note that the lack of statistical significance does not mean that the averages do not differ. A two percentage points difference in the averages, as found in this experiment, may or may not be significant, depending how the researcher needs to use it. In addition, we used an “unweighted average” in which each program is weighted equally in the final mean. If other statistics are used, for instance a weighted mean, the results may differ slightly. The weighted mean of the effectiveness can be computed by summing up the total number of killed mutants on all 39 programs and dividing that sum by the total number of non equivalent mutants.

We did not vary the number of tests; we always used 10. So we are not able to provide insight as to how many test sets are "enough."

Analyzing these results raises an interesting question that might be of interest for researchers and practitioners. What are the consequences of choosing a given test case in terms of effectiveness, considering not only mutation operators but also real faults? If a tester is presented with two test sets adequate to a set of mutants (or to any other testing criterion), which should she/he choose? Is there a way to predict which should be more effective in terms of a larger set of faults, either those defined by mutant operators or real faults? In a quick check we could verify, for instance, that there is no strong correlation between the size of the SSDL-adequate test sets and their effectiveness against the complete set of mutants.

Acknowledgments

Prof. Marcio Delamaro's research is supported by FAPESP (Fundação de Amparo a Pesquisa do Estado de São Paulo), process number 2012/16950-5.

References

[1] M. E. Delamaro, V. H. S. Durelli, L. Deng, N. Li, and J. Offutt, "An empirical evaluation of SDL and one-op mutation," in *International Conference on Software Testing, Verification and Validation (ICST 2014)*, Cleveland, US, 2014.

[2] P. G. Frankl, S. N. Weiss, and C. Hu, "All-uses versus mutation testing: An experimental comparison of effectiveness," *Journal of Systems and Software, Elsevier*, vol. 38, no. 3, pp. 235–253, 1997.

[3] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand, "Experiments on the effectiveness of dataflow- and controlflow-based test adequacy criteria," in *Proceedings of the Sixteenth International Conference on Software Engineering*. Sorrento, Italy: IEEE Computer Society Press, May 1994, pp. 191–200.

[4] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge, UK: Cambridge University Press, 2008, iSBN 0-52188-038-1.

[5] M. E. Delamaro and J. C. Maldonado, "Proteum-A tool for the assessment of test adequacy for C programs," in *Proceedings of the Conference on Performability in Computing Systems (PCS 96)*, New Brunswick, NJ, July 1996, pp. 79–95.

[6] H. Agrawal, R. DeMillo, R. Hathaway, W. Hsu, W. Hsu, E. Krauser, R. J. Martin, A. Mathur, and G. Spafford, "Design of mutant operators for the C programming language," Software Engineering Research Center, Purdue University, West Lafayette IN, Technical Report SERC-TR-41-P, March 1989.

[7] R. Untch, "On reduced neighborhood mutation analysis using a single mutagenic operator," in *ACM Southeast Regional Conference*, Clemson SC, March 2009, pp. 19–21.

[8] L. Deng, J. Offutt, and N. Li, "Empirical evaluation of the statement deletion mutation operator," in *6th IEEE International Conference on Software Testing, Verification and Validation (ICST 2013)*, Luxembourg, March 2013.

[9] R. Langley, *Practical Statistics Simply Explained*. New York: Dover Publications, 1971.