ISSTA 2008
International Symposium
on Software Testing and Analysis
Seattle, WA, July 20-24 2008

Controlled Redundancy

Avoiding Test Suite Wear-Out


Shin Yoo
King's College London
Strand, London WC2R 2LS, UK
Shin.Yoo@kcl.ac.uk


Mark Harman
King's College London
Strand, London WC2R 2LS, UK
Mark.Harman@kcl.ac.uk


Shmuel Ur
IBM, Haifa Research Lab
Haifa University Campus
Haifa, 31905, Israel
Ur@il.ibm.com

Abstract

It is advantageous for a test suite to have high controlled redundancy; that is, to contain many disjoint minimised subsets, each of which satisfies testing criteria with different test inputs. A highly latent test suite denotes a rich source of test cases from which to select. High controlled redundancy can help to avoid 'test suite wear out', in which iterative testing increasingly applies the same test cases to the same code, thereby revealing less and less with each successive iteration. This paper introduces a theory of controlled redundancy. It empirically explores the relationship between controlled redundancy and fault detection ability and introduces a search-based algorithm for improving controlled redundancy.

1. Introduction

Test suite reduction techniques aim to reduce the size of large test suites by removing the redundant test cases. However, since testing can only show the existence of faults instead the lack of them, redundancy in testing should be encouraged rather than discouraged, if the cost remains manageable [1, 2].

However, if the redundancy of a test suite is simply defined as the amount of unnecessary test cases, increasing redundancy becomes trivial and, more importantly, pointless. For example, suppose there are two separate test suites for a single program, S_1 and S_2. Both S_1 and S_2 contain 100 test cases, and both test suites are redundant in a sense that a subset (gray area) of each achieves the given test objective, which is measured as Q.

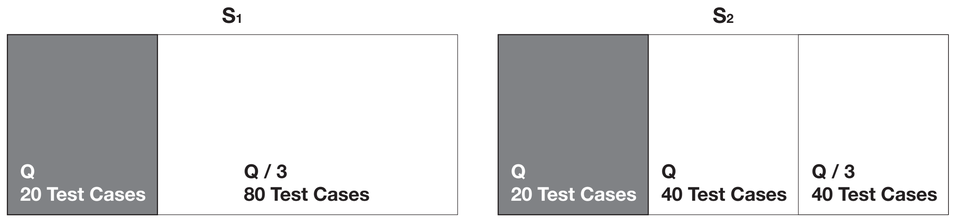

Figure 1. Test suite S_1 and S_2 contains 100 test cases respectively. Both are redundant; it requires only 20 test cases (gray area) to achieve the test objectives, which is measured by Q. However, the redundant parts of two test suites show different quality. Redundant test cases in S_1 only achieves one third of the test objectives, whereas those in S_2 are capable of achieving the test objectives once again, and then another one third achievement. Between these two redundant test suites, S_2 has *better* redundancy.

If we focus on the redundant parts, two test suites show different quality. The redundant test cases in S_1 can only achieve one third of the given test objectives, whereas those in S_2 are capable of achieving it for the second time, and another one third of the test objectives. Between these redundant test suites, the redundancy in S_2 is more *useful*. We call this quality *controlled redundancy*; redundancy in a test suite becomes more useful if the redundant part is guided to achieve the test objective independently. Higher controlled redundancy means a good pool of test cases that can achieve the given test objective in various ways.

2. Definition

Intuitively, controlled redundancy measure is defined as the maximum number of times that a group of disjoint subsets of a test suite can achieve the test objectives. More formally, let T be a test suite with n test cases, $t_1,..., t_n$. Let R be the set of m test requirements, $r_1,..., r_m$. Let $f_R : 2^T \rightarrow \{0, 1\}$ be a function that determines whether a subset of T satisfies the set of test requirements in R; it returns 1 if the subset satisfies R, 0 otherwise. Let P be a partitioning of T, $\{P_1,..., P_k\}$, such that any two members of P are disjoint and $\cup P_i = T$. Finally, let S be the set of all subsets of T that satisfy R respectively: $S = \{T' \mid T' \in 2^T \wedge f_R(T') = 1\}$. Then the controlled redundancy ϱ of T with respect to test requirements R and a partitioning P can be defined as following.

$$\rho = |S \cap P|$$

3. Measurement

We set the test objective to statement coverage and measure controlled redundancy by repeatedly applying test suite reduction algorithm [3]. Each iteration of the algorithm takes the remaining test cases and reduces it to the next partition. For each iteration, the statement coverage achieved by each partition is measured.

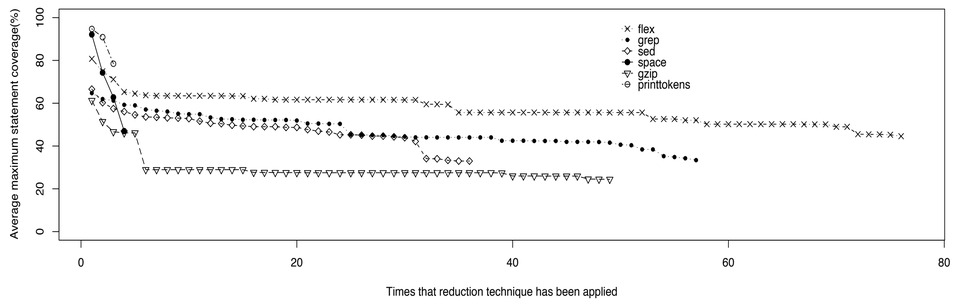

Figure 2. Greedy test case reduction technique is repeatedly applied to test suites of programs in Software Infrastructure Repository (SIR) [4]. Notice that only 4 subsets are possible for the program `space` and that coverage drops dramatically for all test suites considered. Test suites that were generated just to meet the coverage criteria do not provide much redundancy. Test suites generated by utilising program specifications show more redundancy.

Test suites for `space` and `printtokens` are generated so that no more test cases were added once the full coverage is achieved. Naturally, they have little redundancy. Test suites for other programs are generated from their specifications, resulting in higher redundancy in structural coverage.

4. Enhancement

In order to enhance the controlled redundancy of a test suite, the test suite must be augmented with additional test cases that help the achievement of test objectives. We use a search-based test data augmentation technique in order to generate these additional test cases with low cost [5]. Random test data generation has been used to produce test suites with *un*controlled redundancy, using the same amount of computational resource as the search-based approach. The effectiveness of the augmented test suites is evaluated by measuring structural coverage and mutation score.

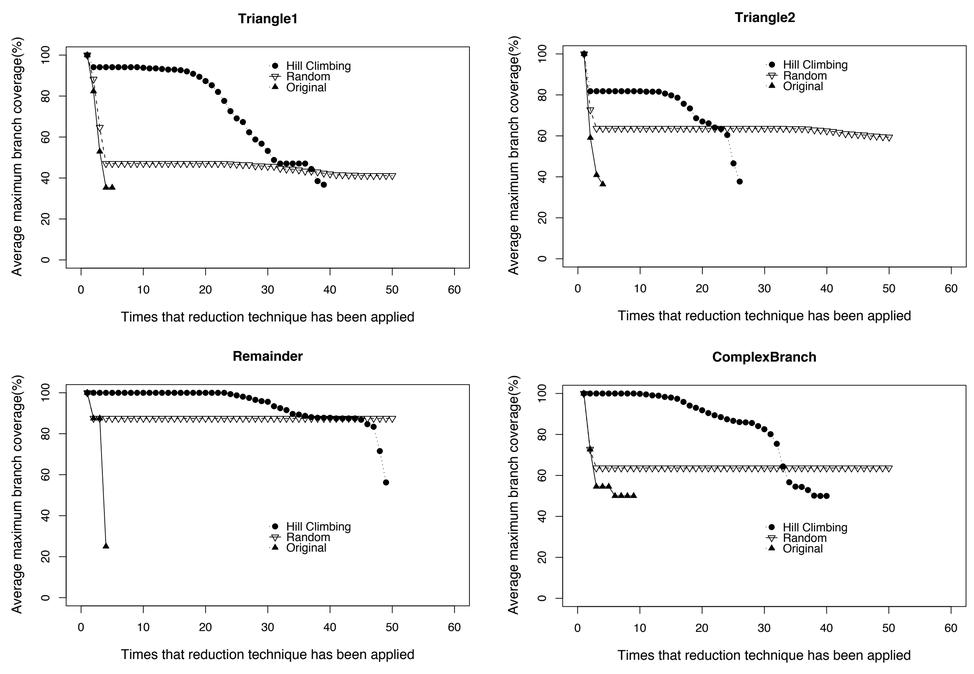

Figure 3. Comparison between the branch coverage from the original test suites and the augmented test suites (Hill Climbing). After applying test data augmentation, coverage drops much slower (`triangle1`, `triangle2`) or remains at 100% (`remainder`, `complexbranch`).

Results in Figure 3 show that after enhancing controlled redundancy, test suites can maintain over 80% coverage for several number of partitions. The lack of full coverage observed in `triangle1` and `triangle2` is due to the specific behaviour of the test data augmentation algorithm used in the experiment. More recent version of the same algorithm now can achieve 100% coverage for these programs.

Enhanced controlled redundancy has a positive impact on mutation score. For all programs except `triangle2`, the test suites with enhanced controlled redundancy show increase in mutation score ranging from 3 to 30. For `triangle2`, the lack of full coverage had a negative impact on mutation score.

5. Conclusions

Redundancy can be measured and controlled with respect to specific test objectives. If a test suite is redundant with respect to a specific test objective, it is possible to achieve the test objective in more than one way, which can increase the quality of testing. When a given test suite is not redundant enough, test data augmentation can generate additional test data from the existing ones with less cost than test data generation.

References

1. G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In ICSM, pages 34–43, 1998.

2. G. Rothermel, M. Harrold, J. Ronne, and C. Hong. Empirical studies of test suite reduction. Journal of Software Testing, Verification, and Reliability, 4(2), December 2002.

3. M. J. Harrold, R. Gupta, and M. L. Soffa. A methodology for controlling the size of a test suite. ACM Trans. Softw. Eng. Methodol., 2(3):270–285, 1993.

4. H. Do, S. G. Elbaum, and G. Rothermel. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering: An International Journal, 10(4):405–435, 2005.

5. S. Yoo and M. Harman. Test data augmentation : generating new test data from existing test data. Technical Report TR-08-04, King's College London, 2008.