Science of Computer Programming ••• (••••) •••–•••

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

An experimental comparison of edge, edge-pair, and prime path criteria

Vinicius H.S. Durelli [a,*], Marcio E. Delamaro [b], Jeff Offutt [c]

[a] Department of Computer Science, Federal University of São João del-Rei, São João del-Rei, Minas Gerais, Brazil
[b] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, São Paulo, Brazil
[c] Software Engineering, George Mason University, Fairfax, VA, USA

ARTICLE INFO

Article history:
Received 27 March 2016
Received in revised form 5 August 2017
Accepted 6 October 2017
Available online xxxx

Keywords:
Structural testing
Prime path coverage
Edge pair coverage
Mutation testing

ABSTRACT

Background: Many criteria have been proposed to generate test inputs. Criteria are usually compared in terms of subsumption: if a criterion C1 *subsumes* C2, it is guaranteed that every test set that satisfies C1 will also satisfy C2. An implication of this notion of subsumption is that C1-adequate tests tend to find more faults than C2-adequate tests, but C1-adequate tests tend to be larger. Thus, while useful, the idea of subsumption does not elaborate on some practical properties of expensive criteria as, for instance, how many more faults a C1-adequate test set will find? More generally, what is the return on investment for choosing more expensive criteria?
Method: To provide a more accurate idea of the fault finding ability and cost of several criteria, we set out to compare three structural graph coverage criteria: edge coverage (EC), edge-pair coverage (EPC), and prime path coverage (PPC). PPC and EPC subsume EC. To compare these criteria we examined 189 functions from 39 C programs, used mutants as a proxy for faults, and performed a statistical analysis of the results.
Result: The three criteria are very similar in terms of effectiveness when all mutants are taken into account: PPC killed 98% of the mutants, EPC 97%, and EC 94%. However, the difference between the criteria is emphasized with minimal mutant sets: PPC killed 75% of the mutants, EPC killed 67%, and EC killed only 57%. As for the cost of these criteria, we found that there is not much difference in terms of the number of TRs. We expected PPC to have the most TRs, so we were surprised to find that, on average, the number of TRs for EPC was highest.
Conclusion: PPC can detect more faults, specially in programs that have complicated control flows, but at higher cost. Thus, a practical tester can make an informed cost versus benefit decision. A better understanding of which structures in the programs contribute to the expense might help to choose when to use PPC.

1. Introduction

Many software test criteria have been defined, and their use in industry is expanding. Generally, *test criteria* apply engineering rules to source code or other software artifacts to create test requirements. A *test requirement* is a specific element of a software artifact that a test case must satisfy or cover. For example, the criterion *edge coverage* creates a test require-

* Corresponding author.
E-mail addresses: durelli@ufsj.edu.br (V.H.S. Durelli), delamaro@icmc.usp.br (M.E. Delamaro), offutt@gmu.edu (J. Offutt).

https://doi.org/10.1016/j.scico.2017.10.003
0167-6423/

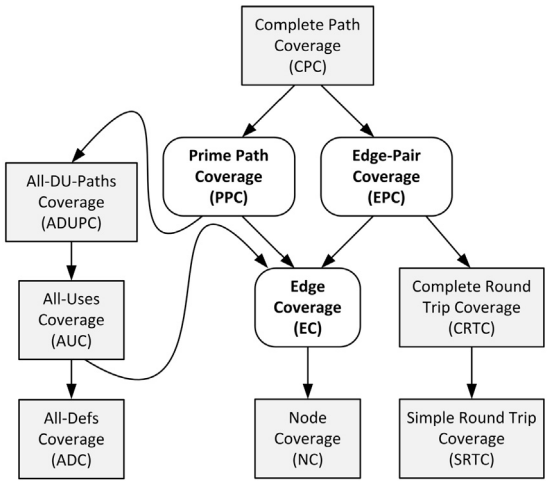

Fig. 1. Subsumption relationships among structural coverage criteria. (*Adapted from Introduction to Software Testing, by Ammann & Offutt, used with permission.*)

ment for each edge in a graph. One of the more common bases for defining test criteria on source code is the control flow graph (CFG). There, edge coverage requires that each edge in a CFG be covered, that is, each branch must resolve to both true and false (hence edge coverage on CFGs is also called *branch coverage*). Because the CFG summarizes the structure of software, these are called *structural criteria*. Ammann & Offutt's textbook [1] define ten structural criteria. With this many choices, practitioners, educators, and researchers need to know which criterion is best for specific situations.

Test criteria can be compared both theoretically and experimentally. The two most common theoretical comparison techniques are subsumption and the number of test requirements. A criterion C1 *subsumes* another criterion C2 if and only if every test set that satisfies C1 is guaranteed to satisfy C2. For example, if a test set takes every branch in a CFG, that test set is guaranteed to cover every node, thus edge coverage subsumes node coverage (also called *statement coverage*). Researchers also analyze test criteria to compute theoretical bounds on the number of test requirements. For example, edge coverage yields $\mathcal{O}(E)$ test requirements, where E is the number of edges in the graph. Although this also bounds the number of test cases needed, some tests cover many test requirements so, in practice, the number of tests is often much less than the number of test requirements.

The most common two experimental comparison techniques are to count the number of actual test requirements (an estimation of cost) and to count the number of faults found by test sets that satisfy the requirements (an estimation of effectiveness). Faults for experimental test subjects (programs, classes, or methods) are commonly obtained in three ways. One is to use naturally occurring faults. Although intuitively satisfying because finding "real faults" is ultimately what we care about, it is often hard to obtain naturally occurring faults, and very hard to obtain them in numbers enough to lend any credibility to our experiments. Another method is to manually seed faults. This is also problematic because it is prone to human error and bias. The person seeding the faults not only cannot be the same person who designs tests, it is much better if the fault seeder does not know the criteria being compared, or even know the purpose of the experiment. A third, and the most common in the research literature, is to create pseudo-faults with mutation. *Mutation analysis* creates many versions of the software, each one differing from the original by one small syntactic change. Mutants can be created in great numbers and are created systematically, in particular, many types of faults are created. Mutation has also been shown to be an excellent proxy for real faults by Andrews et al. [2] and Just et al. [3].

The ten structural criteria defined in Ammann & Offutt's textbook [1] are shown in Fig. 1, with edges representing theoretical subsumption relationships. Although the subsumption relationships are known, the literature offers little in terms of experimental comparisons. This paper compares the criteria that are not highlighted in Fig. 1. Complete path coverage yields an infinite number of test requirements, so is omitted as being not practical. The two round trip criteria are not used in practice, partly because they do not even ensure that all statements (nodes) are covered. Thus, we omit them as well. Our current study is limited to the CFG without including additional information that is needed for data flow, so the data flow criteria are out of scope for this paper. We omit node coverage (*NC*) because it is very similar to edge coverage, both theoretically and in practice. Thus, this paper compares edge coverage (*EC*), edge-pair coverage (*EPC*), and prime path coverage (*PPC*) (these are defined fully in Section 2).

The subsumption relationships among these three criteria are well-known. We also know the bounds on the number of test requirements for *EC* (specifically, $\mathcal{O}(E)$, where E is the number of edges). Nobody has yet published the bounds on the number of edge-pair requirements. Also not known are the cost and effectiveness tradeoffs among *EC*, *EPC*, and *PPC*. This paper reports on an attempt to fill those gaps by supplying theoretical bounds on the number of EPC requirements and presenting data from an empirical comparison of the three structural coverage criteria.

We compared the criteria on 189 functions in 39 C programs. As in much of the test criteria literature, we used mutants as proxies for faults. We also added one more variable to our experiment by comparing not just the percentage of all

Please cite this article in press as: V.H.S. Durelli et al., An experimental comparison of edge, edge-pair, and prime path criteria, Sci. Comput. Program. (2017), https://doi.org/10.1016/j.scico.2017.10.003