Empirical Study on Measuring the t-wise Coverage of $(t-1)$-wise Covering Array

Submission ID: #1171

I. Setup of Empirical Study

We conduct an empirical study to analyze whether a $(t-1)$-wise covering array (CA) covers the majority of valid t-wise tuples, recalling that a $(t-1)$-wise CA is a test suite covering all valid $(t-1)$-wise tuples. In this empirical study, we adopt a standard metric called t-wise coverage to assess the number of valid t-wise tuples covered by a given test suite [1].

Given an SUT S, its test suite E and a testing strength t, the t-wise coverage of E is the ratio between the number of E's covered valid t-wise tuples, and the number of all valid t-wise tuples for S. It is clear that if a $(t-1)$-wise CA achieves higher t-wise coverage, then it covers a larger number of t-wise tuples. To this end, the target of the empirical study is equivalent to analyzing whether a $(t-1)$-wise CA could obtain high t-wise coverage in practice.

For the empirical study, we adopt a benchmarking of 5 public instances collected from real-world, highly configurable systems. These instance include `apache`, a well-known open-source web server application; `bugzilla`, a prominent bug-tracking system; `gcc`, a widely-used compiler collection; and `spins` and `spinv`, the simulator and verifier of SPIN, a widely-used model checker. For more information about these instances, readers can refer to the literature [2].

Since *AutoCCAG* [3] is a state-of-the-art algorithm for solving the t-wise CCAG problem, we employ *AutoCCAG* in our empirical study to generate $(t-1)$-wise CAs. Since *AutoCCAG* is a randomized algorithm [3], for *AutoCCAG* we perform 10 independent runs per instance, and the cutoff time for each algorithm run is set to 1,000 seconds. We note that since *AutoCCAG* fail to generate 4-wise CAs for two instances called `apache` and `gcc` within the cutoff time of 1,000 seconds, for these two instances the cutoff time for each algorithm run is set to 10,000 seconds. Particularly, in this empirical study, we use *AutoCCAG* to generate 2-wise, 3-wise and 4-wise CAs for each instance. That is, in this empirical study we investigate whether 2-wise, 3-wise and 4-wise CAs can achieve high 3-wise, 4-wise and 5-wise coverage, respectively.

II. Experimental Results

In this empirical study, for each instance, we use '3-wise #Cov.' to denote the average number of valid 3-wise tuples covered by *AutoCCAG*'s generated 2-wise CA, '3-wise #Tot.' to represent the total number of valid 3-wise tuples, and '3-wise coverage' to stand for the average 3-wise coverage achieved by *AutoCCAG*'s 2-wise CA. Also, for each instance, we use '4-wise #Cov.' to denote the average number of valid 4-wise tuples covered by *AutoCCAG*'s generated 3-wise CA, '4-wise #Tot.' to represent the total number of valid 4-wise tuples, and '4-wise coverage' to stand for the average 4-wise coverage achieved by *AutoCCAG*'s 3-wise CA. Similarly, for each instance, we use '5-wise #Cov.' to denote the average number of valid 5-wise tuples covered by *AutoCCAG*'s generated 4-wise CA, '5-wise #Tot.' to represent the total number of valid 5-wise tuples, and '5-wise coverage' to stand for the average 5-wise coverage achieved by *AutoCCAG*'s 4-wise CA.

TABLE I
Results of *AutoCCAG*'s generated t-wise CAs on all 5 instances.

Instance	2-wise CA		
	3-wise #Cov.	3-wise #Tot.	3-wise coverage
`apache`	7,649,774.5	8,085,958	94.6%
`bugzilla`	175,404.4	202,683	86.5%
`gcc`	9,879,399.2	11,131,894	88.7%
`spins`	9,124.1	12,835	71.1%
`spinv`	305,812.1	369,976	82.7%
Instance	**3-wise CA**		
	4-wise #Cov.	4-wise #Tot.	4-wise coverage
`apache`	720,948,788.8	728,107,664	99.0%
`bugzilla`	4,840,264.6	5,182,503	93.4%
`gcc`	1,100,152,057.7	1,116,587,521	98.5%
`spins`	96,677.2	116,332	83.1%
`spinv`	10,782,514.0	11,427,629	94.4%
Instance	**4-wise CA**		
	5-wise #Cov.	5-wise #Tot.	5-wise coverage
`apache`	52,067,979,118.1	52,120,536,127	99.9%
`bugzilla`	101,582,755.6	103,702,986	98.0%
`gcc`	80,115,075,466.4	89,097,266,296	89.9%
`spins`	702,470.4	774,940	90.6%
`spinv`	270,221,907.1	274,659,327	98.4%

The related experimental results are reported in Table I. According our results in Table I, it is clear that $(t-1)$-wise CA can achieve high t-wise coverage, indicating that $(t-1)$-wise CA is able to cover the majority of valid t-wise tuples.

References

[1] C. Luo, B. Sun, B. Qiao, J. Chen, H. Zhang, J. Lin, Q. Lin, and D. Zhang, "LS-Sampling: An effective local search based sampling approach for achieving high t-wise coverage," in *Proceedings of ESEC/FSE 2021*, 2021, pp. 1081–1092.

[2] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "An improved meta-heuristic search for constrained interaction testing," in *Proceedings of International Symposium on Search Based Software Engineering 2009*, 2009, pp. 13–22.

[3] C. Luo, J. Lin, S. Cai, X. Chen, B. He, B. Qiao, P. Zhao, Q. Lin, H. Zhang, W. Wu, S. Rajmohan, and D. Zhang, "AutoCCAG: An automated approach to constrained covering array generation," in *Proceedings of ICSE 2021*, 2021, pp. 201–212.