Study on t-wise Coverage

RQ: Can 3-wise CA constructed by *ScalableCA* achieve high t-wise coverage ($4 \le t \le 6$)?

Empirical studies [1–6] indicate that a certain number of faults are triggered by the combinations of 4 to 6 options, so a test suite of high t-wise coverage ($4 \le t \le 6$) can detect more faults. In this RQ, we perform evaluations to study whether *ScalableCA*'s built 3-wise CA could obtain high t-wise coverage ($4 \le t \le 6$).

1 EXPERIMENTAL RESULTS

For testing highly configurable systems, in practice a certain number of faults are known to be caused by the combination of 4 to 6 options [1–6]. Hence, given a test suite, if it could obtain higher t-wise coverage ($4 \le t \le 6$), then more faults could be disclosed. In this subsection, we empirically analyze the t-wise coverage ($4 \le t \le 6$) achieved by 3-wise CAs and 2-wise CAs. According to the definition of t-wise coverage in Section 2.1 of the paper, calculating the exact t-wise coverage needs to obtain the number of all valid t-wise tuples for a given configurable system. However, due to the existence of huge numbers of valid t-wise tuples ($4 \le t \le 6$) for highly configurable systems, it is impractical to enumerate all of them, so calculating the exact t-wise coverage ($4 \le t \le 6$) is infeasible. To mitigate this issue, following a recent study on t-wise coverage [7], in our experiments we estimate the t-wise coverage as follows: given a test suite T (*i.e.,* a 2-wise CA or a 3-wise CA in our experiments), for each $t \in \{4, 5, 6\}$, we first construct an estimation set E containing 10^7 uniformly sampled, valid t-wise tuples per instance, then the t-wise coverage of T is estimated as the number of those valid t-wise tuples, which belong to E and are meanwhile covered by T, divided by E's cardinality (*i.e.,* $|E| = 10^7$).

Table 1 reports the average t-wise coverage ($4 \le t \le 6$) of the 3-wise CAs by *ScalableCA*, *SamplingCA* and *CAmpactor*, as well as the 2-wise CAs by *SamplingCA* and *CAmpactor*. Table 1 shows that 3-wise CA achieves much higher t-wise coverage ($4 \le t \le 6$) than 2-wise CA, indicating the superiority of 3-wise CIT over 2-wise CIT, and confirming the importance of developing effective approaches for generating 3-wise CA. Also, existing empirical studies on various real-world, highly configurable systems [1–4] present that in practice a test suite with high t-wise coverage ($4 \le t \le 6$) could detect the majority of faults. According to Sections 6.1 and 6.2 of the paper, our results show that *ScalableCA* can generate 3-wise CA to achieve similar high t-wise coverage ($4 \le t \le 6$) as our competitors do, but with significantly smaller size and much less running time. Additionally, as Section 7.2 of the paper, the 3-wise CAs generated by *ScalableCA* exhibit a higher fault detection rate compared to its competitors, indicating that adopting *ScalableCA* could bring much benefit in real-world applications.

Table 1: Average t-wise coverage ($4 \le t \le 6$) of 3-wise CAs by *ScalableCA*, *SamplingCA* and *CAmpactor*, as well as 2-wise CAs by *SamplingCA* and *CAmpactor*. To save space, we use notation 'cov.' to denote 'coverage'.

	4-wise cov.	5-wise cov.	6-wise cov.
ScalableCA's 3-wise CA	99.9%	98.9%	95.5%
SamplingCA's 3-wise CA	99.9%	99.5%	97.5%
CAmpactor's 3-wise CA	99.9%	99.3%	97.0%
SamplingCA's 2-wise CA	95.5%	86.6%	72.7%
CAmpactor's 2-wise CA	87.6%	71.8%	54.1%

REFERENCES

[1] D. Richard Kuhn, Dolores R. Wallace, and Albert M. Gallo. 2004. Software Fault Interactions and Implications for Software Testing. *IEEE Transactions on Software Engineering* 30, 6 (2004), 418–421.

[2] Rick Kuhn and Raghu Kacker. 2011. Practical combinatorial (t-way) methods for detecting complex faults in regression testing. In *Proceedings of ICSM 2011*. 599.

[3] Rick Kuhn, Raghu Kacker, Yu Lei, and Justin Hunter. 2009. Combinatorial Software Testing. *Computer* 42, 8 (2009), 94–96.

[4] Rick Kuhn, Yu Lei, and Raghu Kacker. 2008. Practical Combinatorial Testing: Beyond Pairwise. *IT Professional* 10, 3 (2008), 19–23.

[5] Jinkun Lin, Shaowei Cai, Chuan Luo, Qingwei Lin, and Hongyu Zhang. 2019. Towards More Efficient Meta-heuristic Algorithms for Combinatorial Test Generation. In *Proceedings of ESEC/FSE 2019*. 212–222.

[6] Chuan Luo, Jinkun Lin, Shaowei Cai, Xin Chen, Bing He, Bo Qiao, Pu Zhao, Qingwei Lin, Hongyu Zhang, Wei Wu, Saravanakumar Rajmohan, and Dongmei Zhang. 2021. AutoCCAG: An Automated Approach to Constrained Covering Array Generation. In *Proceedings of ICSE 2021*. 201–212.

[7] Yi Xiang, Han Huang, Miqing Li, Sizhe Li, and Xiaowei Yang. 2022. Looking For Novelty in Search-Based Software Product Line Testing. *IEEE Transactions on Software Engineering* 48, 7 (2022), 2317–2338.