CAopt: Tool Description

Qiyuan Zhao*, Chuan Luo[†✉], Jianping Song[†], Shaowei Cai[‡], Chunming Hu[†]
*School of Computing, National University of Singapore, Singapore
[†]School of Software, Beihang University, China
[‡]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China

***Abstract*—This tool description briefly introduces our tool called *CAopt* for combinatorial interaction testing.**

I. Introduction

Combinatorial interaction testing (CIT) aims to construct a test suite consisting of an acceptable number of test cases, where each test case is sampled from the entire configuration space. In practice, adopting such test suite is able to save a considerable amount of testing effort. Given a configurable system, a t-wise tuple is a combination of values of exactly t options, and the primary goal of t-wise CIT is to generate a t-wise covering array (CA), which is a set of test cases, such that all possible t-wise tuples are covered.

In this tool description, we propose a tool dubbed *CAopt* for generating t-wise CAs. Particularly, *CAopt* consists of two phases, *i.e.,* sampling phase and optimization phase.

II. Sampling Phase of *CAopt*

In the sampling phase, *CAopt* first adopts an efficient sampling method to generate a test suite T to cover valid t-wise tuples as more as possible. Specifically, in the sampling phase *CAopt* constructs a test suite with high t-wise coverage in an iterative manner. In each iteration, *CAopt* samples a candidate collection of test cases that are not similar to existing test cases in T, and then chooses the best test case β^* if it is added into T; that is, $T \cup \{\beta^*\}$ covers the most valid t-wise tuples. Through this way, a test case set T with high t-wise coverage is generated. Then, *CAopt* adds a number of test cases to cover the remaining valid t-wise tuples, in order to make T become a t-wise CA. For the more technical details about the sampling phase of *CAopt*, readers can refer to the literature [1].

III. Optimization Phase of *CAopt*

Once the sampling phase of *CAopt* terminates, *CAopt* steps into the optimization phase. In the optimization phase, *CAopt* adopts a two-mode local search framework [2] to optimize the t-wise CA from the sampling phase. The two-mode local search framework works between greedy mode and random mode. In the greedy mode, *CAopt* aims to seek for a smaller-sized t-wise CA, while in the random mode *CAopt* tends to diversify the search so as to explore promising search space. In the optimization phase, given a t-wise CA of size λ, *CAopt* attempts to find a t-wise CA of size $\lambda - 1$. If this attempt succeeds, then *CAopt* turns to searching a t-wise CA with a further smaller size (*e.g.,*, $\lambda - 2$). Through repeating this procedure, *CAopt* is able to output an optimized t-wsie CA of smaller size.

IV. Input format of *CAopt*

Our *CAopt* tool only supports input in the CTWedge format.

References

[1] C. Luo, Q. Zhao, S. Cai, H. Zhang, and C. Hu, "SamplingCA: Effective and efficient sampling-based pairwise testing for highly configurable software systems," in *Proceedings of ESEC/FSE 2022*, 2022, pp. 1185–1197.

[2] C. Luo, S. Cai, W. Wu, Z. Jie, and K. Su, "CCLS: An efficient local search algorithm for weighted maximum satisfiability," *IEEE Transactions on Computers*, vol. 64, no. 7, pp. 1830–1843, 2015.

✉ Chuan Luo is the corresponding author (Email: chuanluo@buaa.edu.cn).