CAgen: Multithreaded FIPOG Algorithms for Combinatorial Test Generation

Michael Wagner, Manuel Leithner and Dimitris E. Simos
SBA Research
A-1040 Vienna, Austria
{mwagner,mleithner,dsimos}@sba-research.org

Rick Kuhn and Raghu Kacker
NIST, Information Technology Laboratory,
Gaithersburg, MD, USA
Email: {d.kuhn,raghu.kacker}@nist.gov

***Abstract*—This document describes the tool CAgen [1] as submitted for the CT competition of IWCT 2023.**

***Index Terms*—Covering Array, Combinatorial Testing, In-Parameter-Order, Algorithms, Multithreading**

I. Tool overview

CAgen is a state-of-the-art tool that efficiently generates combinatorial test sets using the In-Parameter-Order (IPO) strategy. It implements different FIPOG [2] algorithms, that improve upon classical greedy IPOG algorithms by introducing various improvements on the implementation and algorithmic level. It further supports various constraints, which are handled by a minimal forbidden tuple (MFT) approach. Lastly, a we designed a multithreaded variation of the horizontal extension [3]. This document covers the basic functionality of CAgen, as submitted to the CT competition of IWCT 2023.

II. A multithreaded FIPOG algorithm

The IPO strategy [4] builds a covering array (CA), which is the structure underlying combinatorial test sets, by iteratively adding columns and rows to an initial small CA. In each expansion step, first procedure called horizontal extension adds a new column to the current CA, selecting the best candidate values for each entry in the newly added column in a greedy manner. Afterwards, a vertical extension step ensures coverage of all t-way interactions, adding additional rows if necessary. In [3], three different multithreaded variations of the horizontal extension were introduced. In our tool, we implement the variation *JIT*, where each worker thread manages it's own coverage map and does not conduct any precomuting. Further, we allow the algorithm switch between singlethreading and multithreading on demand and select the optimal number of threads at run time. To determine this, we conducted experiments, constructing uniform CAs with different strengths and logging the run time of each individual horizontal extension step for all possible numbers of threads $1 \dots n$, where n denotes the number of logical processors. The algorithm then decides on the number of threads to use based on the formula that most closely matched our experimental results.

The research presented in this paper has been funded in part by the Austrian COMET K1 program. Moreover, this work was performed partly under the financial assistance award 70NANB21H124 from U.S. Department of Commerce, National Institute of Standards and Technology.

III. Constraint handling

In order to effectively check the validity of each assignment made, CAgen employs a MFT approach. First, the specified constraints are parsed into an abstract syntax tree, from which an initial set of forbidden tuples is derived. Afterwards a set of MFTs is computed, which in addition contain all assignments that are forbidden implicitly. This is required to conduct validation checks of partial assignments during the generation. Since the last CT competition of IWCT 2022, we significantly improved the algorithm that computes the MFTs and implemented constraint handling for numerical constraints.

IV. Input and Output formats

CAgen supports the ACTS input format and outputs the constructed test set in CSV form. Further, it supports all types of constraints in the competition and therefore can participate in all tracks.

V. Results

Table I contains the results for a selected subset of the benchmark instances for strength $2 \dots 4$. All experiments were conducted on a machine equipped with an Intel i7-11700F CPU with 16 logical processors and 16GB of RAM.

References

[1] M. Wagner, K. Kleine, D. E. Simos, R. Kuhn, and R. Kacker, "Cagen: A fast combinatorial test generation tool with support for constraints and higher-index arrays," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2020, pp. 191–200.

[2] K. Kleine and D. E. Simos, "An efficient design and implementation of the in-parameter-order algorithm," *Mathematics in Computer Science*, Dec 2017.

[3] M. Wagner, M. Leithner, D. E. Simos, R. Kuhn, and R. Kacker, "Developing multithreaded techniques and improved constraint handling for the tool cagen," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2022, pp. 87–93.

[4] Y. Lei and K. C. Tai, "In-parameter-order: a test generation strategy for pairwise testing," in *Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231)*, Nov 1998, pp. 254–261.

TABLE I
EXPERIMENTAL RESULTS OF THE mFIPOG ALGORITHM FOR THE BENCHMARK INSTANCES OF THE CT-TOOL COMPETITION OF IWCT 2023.

instance	t	rows	time(ms)
BOOLC_0	2	10	1
	3	26	2
	4	62	4
BOOLC_1	2	10	2
	3	34	3
	4	89	6
BOOLC_2	2	11	1
	3	29	2
	4	77	4
BOOLC_3	2	7	1
	3	14	1
	4	29	1
BOOLC_4	2	4	1
	3	4	1
	4	4	1
MCA_0	2	157	0
	3	1884	12
	4	19769	339
MCA_1	2	81	0
	3	162	0
	4	790	4
MCA_2	2	143	0
	3	1430	0
	4	2860	3
MCA_3	2	185	0
	3	2380	11
	4	25591	237
MCA_4	2	197	0
	3	2117	13
	4	20593	456
CNF_0	2	195	3
	3	2171	30
	4	17828	836
CNF_1	2	110	1
	3	995	5
	4	9032	81
CNF_2	2	168	2
	3	1857	9
	4	14861	178
CNF_3	2	241	3
	3	3673	56
	4	49857	3029
CNF_4	2	536	13
	3	10121	3372
	4	125094	381421