SAND2022-7085C

IDA

Algorithmic Input Generation for More Effective Software Testing

Laura Epifanovskaya, Jinseo R. Lee, Christopher McCormack, Reginald Meeson
Institute for Defense Analyses

Robert C. Armstrong, Jackson R. Mayo
Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Institute for Defense Analyses
4850 Mark Center Drive ● Alexandria, Virginia 22311-1882

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

I. Introduction

The Fourth Industrial Revolution is here: Software powers everything, and software is hard to test

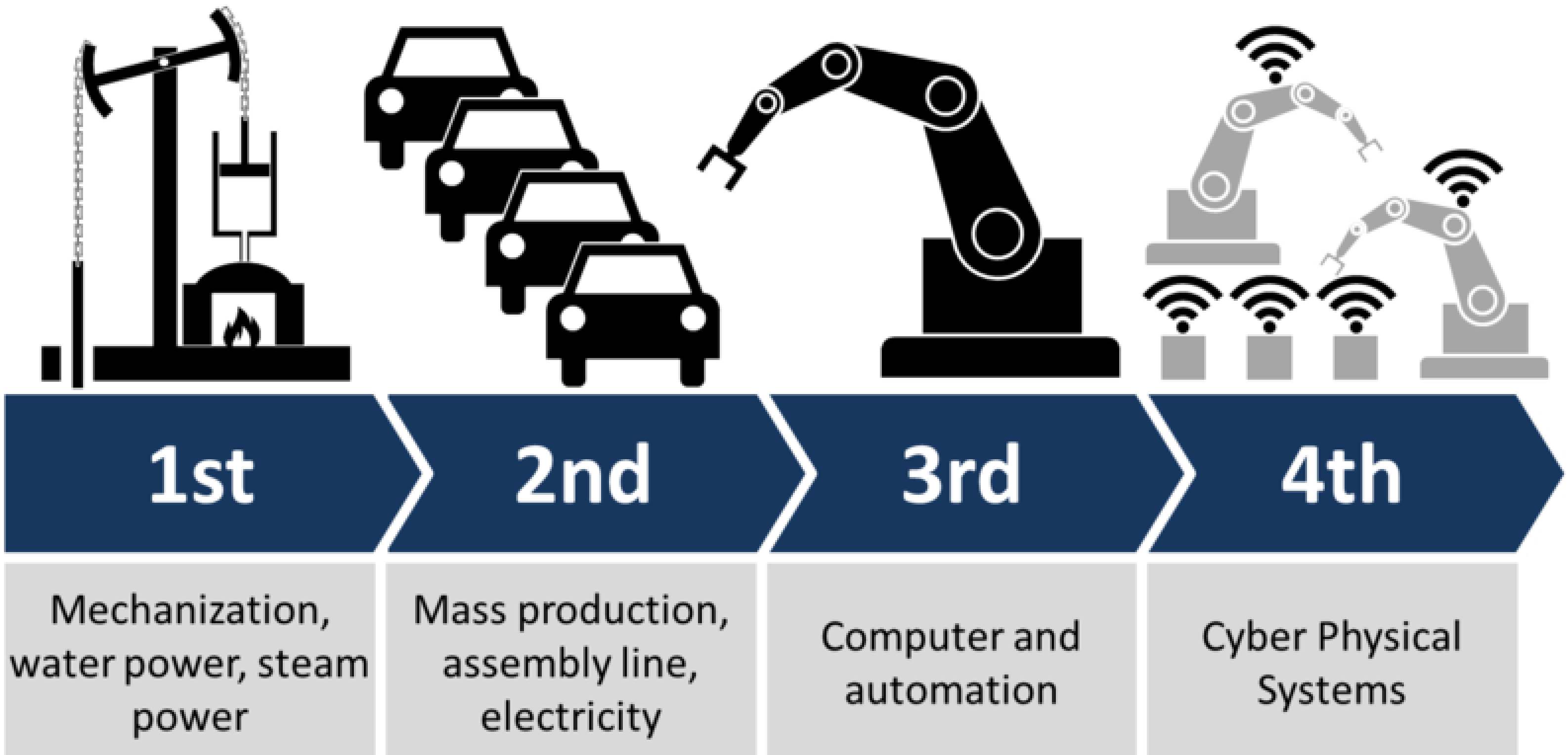


Image: https://www.forbes.com/sites/bernardmarr/2016/04/05/why-everyone-must-get-ready-for-4th-industrial-revolution/?sh=59423e423f90

II. Background

Institute for Defense Analyses
4850 Mark Center Drive • Alexandria, Virginia 22311-1882

Coverage metrics are required but insufficient criteria for testing software

- Modified condition/decision coverage (MC/DC) is required by the standard used in commercial aviation
- Not all variable values are tested
- Masking can undermine the utility of coverage metrics

```
if ((A > 10) && B) {
        C=True
} else {
        C=False
}
```

Condition !(A > 10) is masked if !B

A	B	C
T	T	T
F	T	F
T	F	F
F	F	F

IDA

Ideas from fuzzing suggest ways of sampling a program's input space

- Fuzzing (automated randomized testing) helps find unexpected behaviors
- Rather than purely random inputs, state-of-the-art fuzzing prioritizes "corner cases" and perturbations to normal inputs
- We seek to build on fuzzing practice and target tests to uncover bugs more effectively, by characterizing the input space mathematically

III. Experiment

A small C module of the Traffic Collision Avoidance System (TCAS) used in commercial aviation with 12-variable input and single output

TABLE 1. TCAS VARIABLE VALUES

TCAS Variable	Equivalence Bin Values
Cur_Vertical_Sep	299, 300, 601
High_Confidence	TRUE, FALSE
Two_of_Three_Reports_Valid	TRUE, FALSE
Own_Tracked_Alt	1, 2
Own_Tracked_Alt_Rate	600, 601
Other_Tracked_Alt	1, 2
Alt_Layer_Value	0, 1, 2, 3
Up_Separation	0, 399, 400, 499, 500, 639, 640, 739, 740, 840
Down_Separation	0, 399, 400, 499, 500, 639, 640, 739, 740, 840
Other_RAC	NO_INTENT, DO_NOT_CLIMB, DO_NOT_DESCEND
Other_Capability	TCAS_TA, OTHER
Climb_Inhibit	TRUE, FALSE

28 "buggy" TCAS modules were generated through mutation of the code (changing conditional operators or internal variable values, for example)

This approach replicates work done at the U.S. National Institute of Standards and Technology (NIST) by Richard Kuhn and Vadim Okun‡

‡Kuhn, R. and Vadim Okun, *Pseudo-Exhaustive Testing for Software, Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop SEW-30*

Test inputs were generated using covering arrays, which guarantee t-way variable interactions in a given array

TABLE 1
PARAMETERS FOR PLACING A TELEPHONE CALL

Call Type	*Billing*	*Access*	*Status*
Local	Caller	Loop	Success
Long Distance	Collect	ISDN	Busy
International	800	PBX	Blocked

TABLE 3
PAIR-WISE TEST CASES FOR PLACING A PHONE CALL

Call Type	*Billing*	*Access*	*Status*
Local	Collect	PBX	Busy
Long Distance	800	Loop	Busy
International	Caller	ISDN	Busy
Local	800	ISDN	Blocked
Long Distance	Caller	PBX	Blocked
International	Collect	Loop	Blocked
Local	Caller	Loop	Success
Long Distance	Collect	ISDN	Success
International	800	PBX	Success

Example*:

Nine tests required to include all t=2-way interactions

Full-factorial requires 3^4=81 tests

TABLE 2. T-WAY COVERING ARRAY TEST SETS

Array Strength	Number of Tests
2-way	100
3-way	400
4-way	1215
5-way	3607
6-way	11018

*Cohen, David M., Siddhartha R. Dalal, Michael Freedman, Gardner C. Patton, *The AETG System: An Approach to Testing Based on Combinatorial Design. IEEE Transactions on Software Engineering, 1997.*

IV. Results

Tests generated using covering arrays caught all but one of the program bugs at high t-way interaction levels (t=5, t=6)

TABLE 4. COVERING ARRAY TEST RESULTS

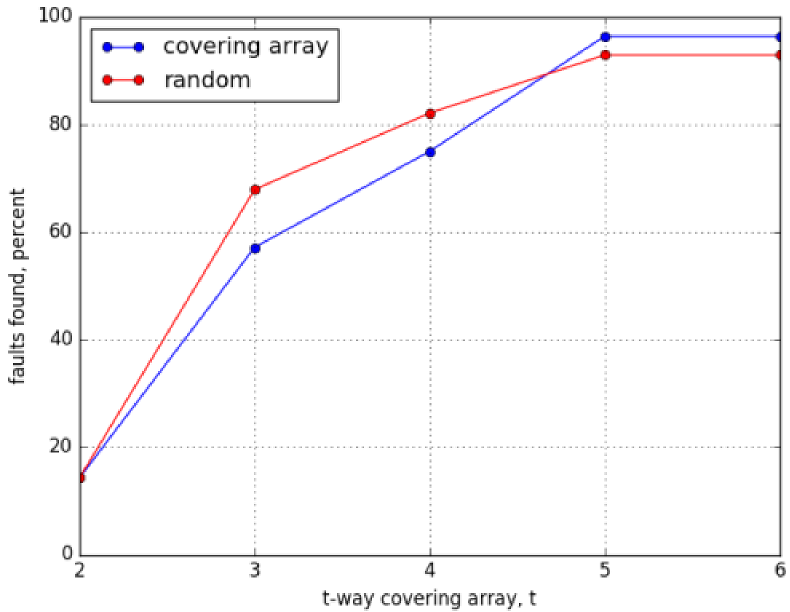
t (strength)	t=2	t=3	t=4	t=5	t=6
Test Size	100	400	1215	3607	11018
Bugs Caught	4	16	21	27	27
Test Failures	103	257	1292	3892	11663
Total Tests	2800	11200	34020	100996	308504
% Efficiency	3.7	2.3	3.8	3.9	3.8

TABLE 1. RANDOM TEST RESULTS

t (strength)	t=2	t=3	t=4	t=5	t=6
Test Size	100	400	1215	3607	11018
Bugs Caught	4	19	23	26	26
Test Failures	78	351	1035	2957	8878
Total Tests	2800	11200	34020	100996	308504
% Efficiency	2.7	3.1	3.0	2.9	2.9

But random test sets of the same size also did well!

Covering arrays do slightly better than random testing with large test sets, but are no better than random at low t-way interactions



The power of covering arrays comes from the forced specification of low-probability interaction sets

$$\frac{1}{10} \times \frac{1}{10} \times \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{2} = \frac{1}{7200}$$

A specific six-way combination has a 78% chance of appearing in a random draw

$$(1 - \frac{1}{7200})^{11016} \sim 0.22$$

The chance is 100% that it will appear in a t=6-way covering array

One fault was never triggered by the covering arrays or random test sets because the binned values did not provide sufficient resolution

Table 1. TCAS Variable Values	
TCAS Variable	Equivalence Bin Values
Cur_Vertical_Sep	299, 300, 601
High_Confidence	TRUE, FALSE
Two_of_Three_Reports_Valid	TRUE, FALSE
Own_Tracked_Alt	1, 2
Own_Tracked_Alt_Rate	600, 601
Other_Tracked_Alt	1, 2
Alt_Layer_Value	0, 1, 2, 3
Up_Separation	0, 399, 400, 499, 500, 639, 640, 739, 740, 840
Down_Separation	0, 399, 400, 499, 500, 639, 640, 739, 740, 840
Other_RAC	NO_INTENT, DO_NOT_CLIMB, DO_NOT_DESCEND
Other_Capability	TCAS_TA, OTHER
Climb_Inhibit	TRUE, FALSE

An internal variable is set to True if Cur_Vertical_Sep = 600 in the correct program

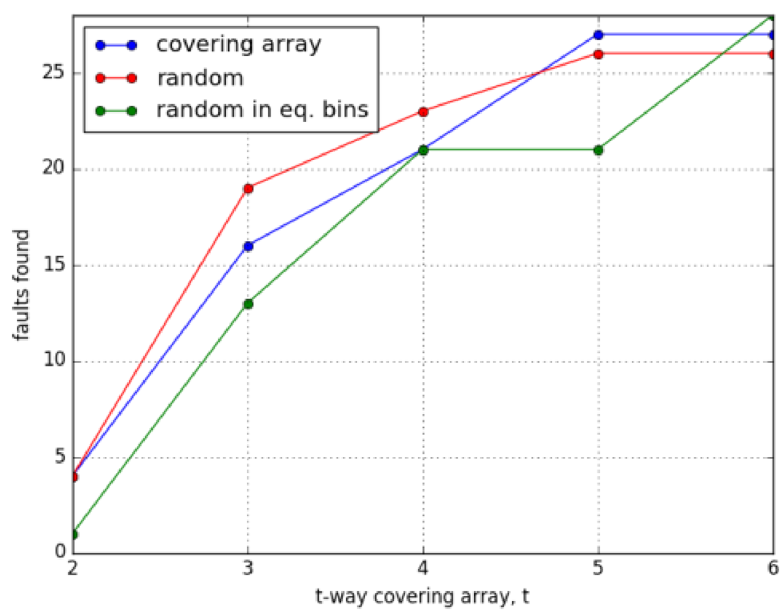
In the faulty program, the logic incorrectly sets the variable to True if Cur_Vertical_Sep = 500

The equivalence binning does not provide resolution to catch the mistake

The problem was overcome by creating covering arrays of randomly select values from the bins (Random from equivalence Bin Covering Array, RBCA)

TABLE 6. RBCA TEST RESULTS

t (strength)	t=2	t=3	t=4	t=5	t=6
Test Size	37	144	476	1334	3837
Bugs Caught	1	13	21.7	23	28
Test Failures	34	154	505	1420	4135
Total Tests	1036	4032	13328	37352	107436
% Efficiency	3.3	3.8	3.8	3.8	3.8



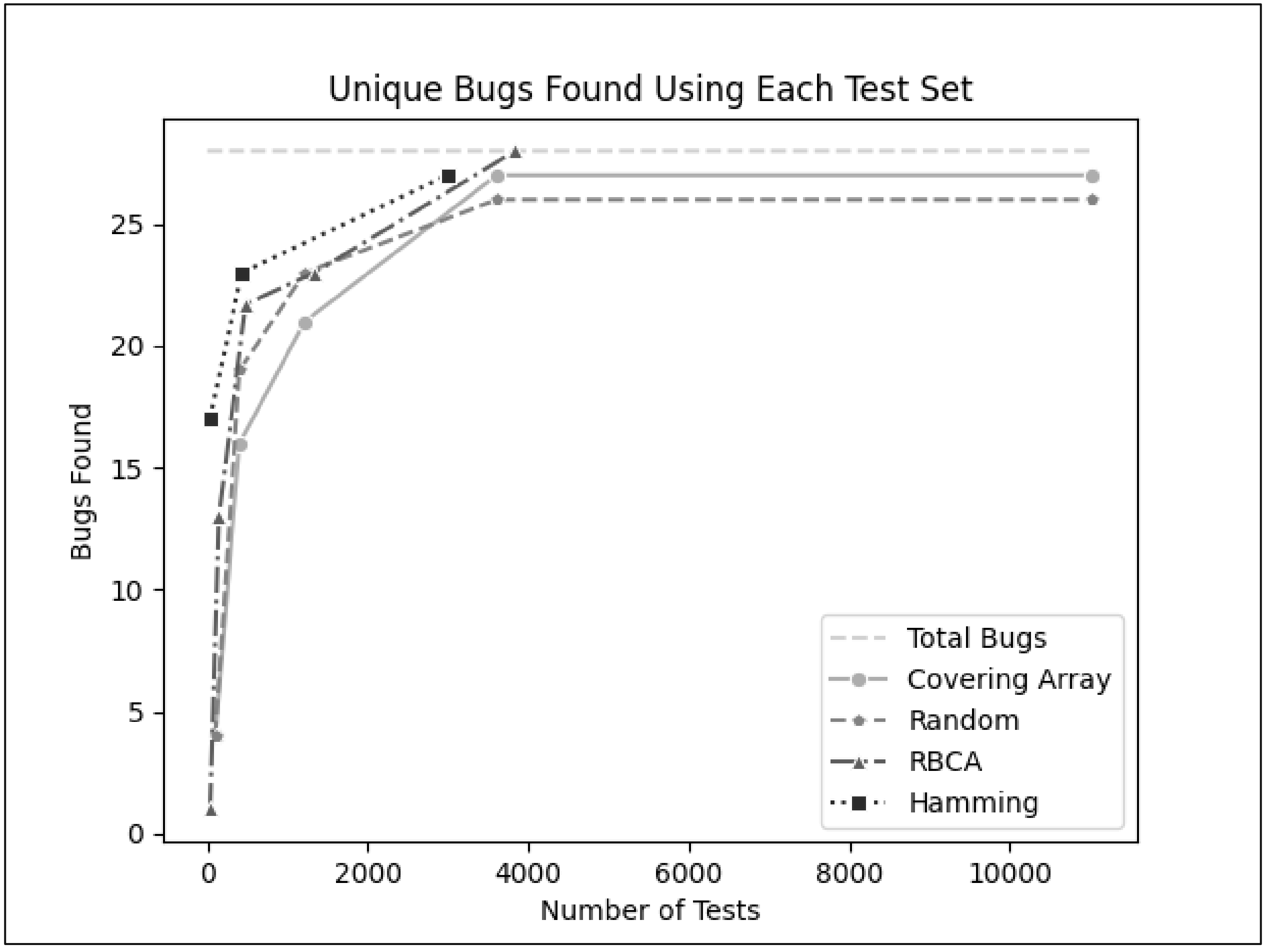
A complexity approach used a single input as a seed, then created a test set based on a specified "Hamming distance" from that seed

Table 7. Hamming Test Results			
	Tier 1	Tier 2	Tier 3
Input Seed Used	Bugs Caught	Bugs Caught	Bugs Caught
UPWARD_RA	13	17	22
UPWARD_RA Tier 1 Output	15	19	22
DOWNWARD_RA	17	22	27
DOWNWARD_RA Tier 1 Output	19	23	27
DOWNWARD_RA Tier 1 Output	17	23	27

Seed input: 299 0 0 2 600 2 0 500 499 0 1 0

Inputs of Hamming distance 1:
299 ***1*** 0 2 600 2 0 500 499 0 1 0
299 0 0 2 600 2 0 500 ***740*** 0 1 0
299 0 0 ***0*** 600 2 0 500 499 0 1 0

The Hamming test sets were more efficient than the others, but also used the equivalence bin values, making one fault unreachable



V. Conclusion

Institute for Defense Analyses
4850 Mark Center Drive ● Alexandria, Virginia 22311-1882

Algorithm-directed fuzzing (Hamming) was the most efficient technique

- If we discard the equivalence bin values and move to a continuum of values, we expect it to catch the faults in all programs
- Our ongoing work is tailoring the fuzzing algorithm and implementing it as a real-time fuzzing tool