***PREVIEW to:***

# Covering Arrays: Evaluating *t*-Coverage & *t*-Diversity in the presence of disallowed combinations

## ITEA 2018

## 6th Cyber Security Workshop

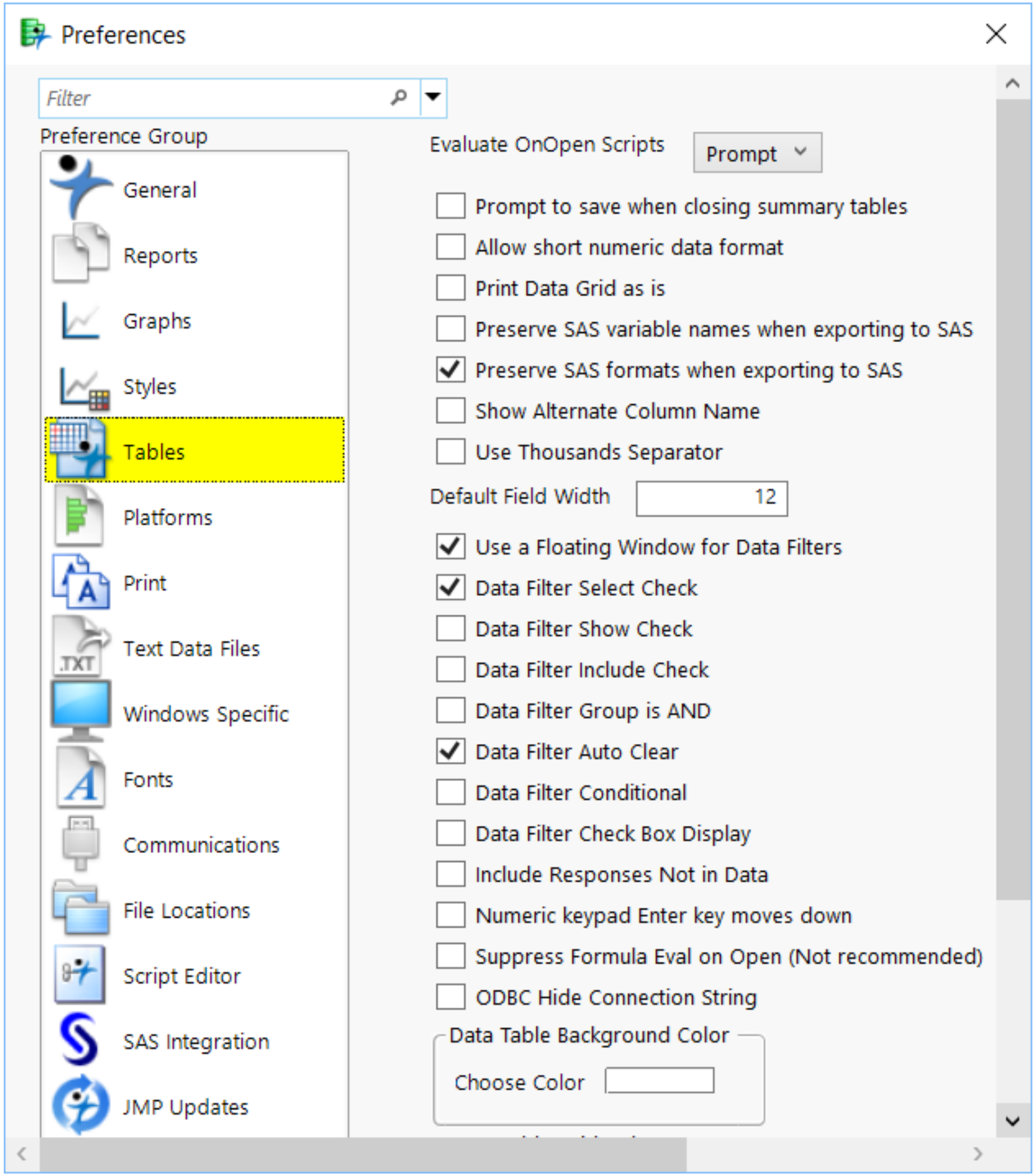
Joseph Morgan, Ryan Lekivetz, & *Tom Donnelly*

JMP Division

SAS Institute Incorporated

Cary, North Carolina 27513

Tom_Donnelly@JMP.com

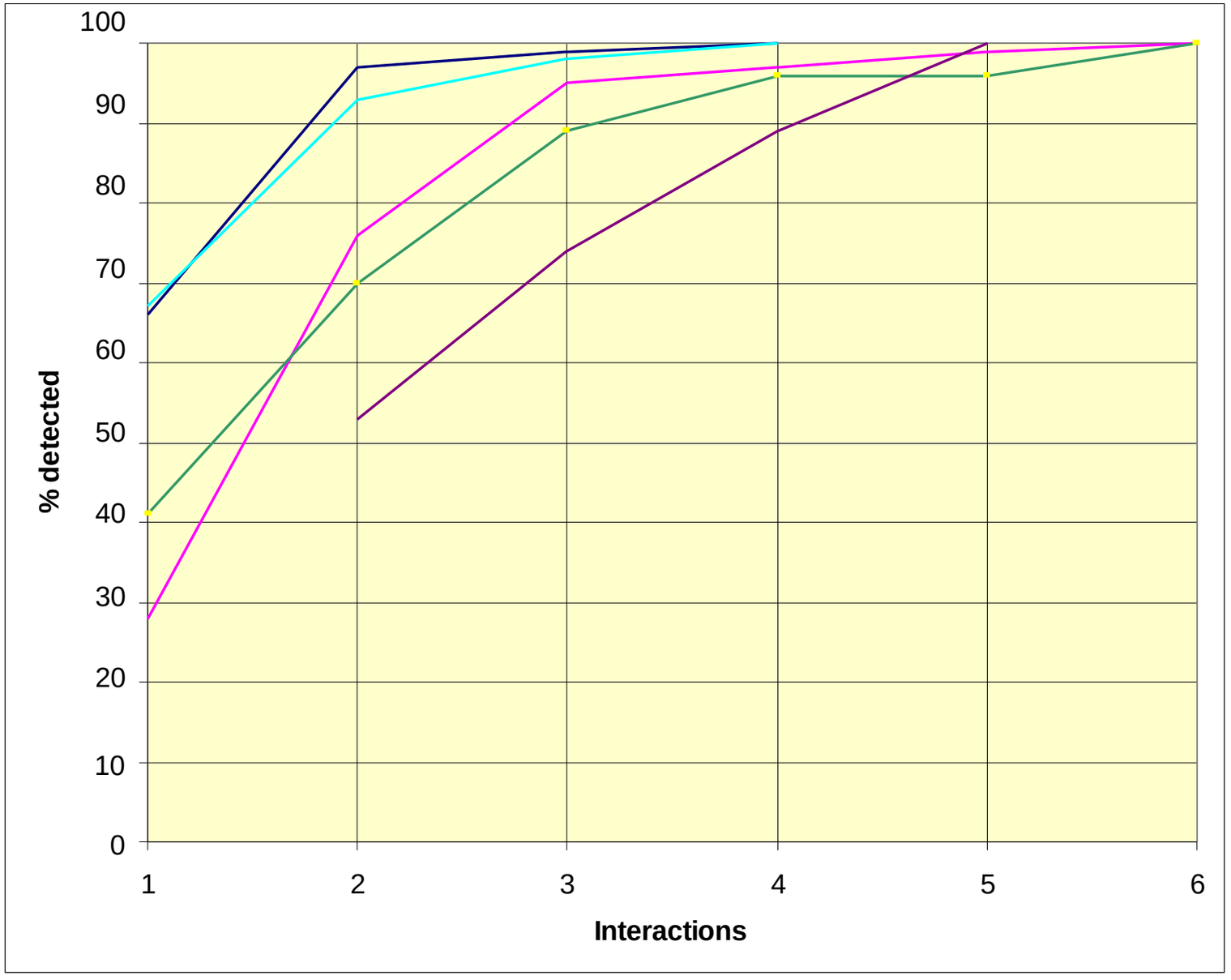



Twenty check boxes in this dialog box

$2^{20}$ = 1,048,576 possible combinations

How many tests to check:
All pairs?
All triples?
All quadruples?
All quintuples?
All sextuples?

# Traffic Collision Avoidance System module (seeded errors) (purple)




Slide courtesy of Rick Kuhn of NIST

# So, how many parameters are involved in really tricky faults?

- Maximum interactions for fault triggering for these applications was 6
- Much more empirical work needed
- Reasonable evidence that maximum interaction strength for fault triggering is relatively small


Slide courtesy of Rick Kuhn of NIST

# All triples in only 18 tests covering 9,120 combinations
# This test matrix is called a covering array of strength 3.

20f 2L Covering Array t3 18 - JMP Pro

File Edit Tables Rows Cols DOE Analyze Graph Six Sigma Tools Tools Add-Ins View Window Help

21/0 Cols

18/0

| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 | X18 | X19 | X20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 |
| 2 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 |
| 3 | L2 | L2 | L2 | L2 | L2 | L2 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | L2 |
| 4 | L2 | L2 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L2 |
| 5 | L2 | L2 | L1 | L2 | L1 | L1 | L2 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L1 | L2 | L1 | L2 | L1 |
| 6 | L1 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L2 | L1 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L2 |
| 7 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L2 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L2 | L1 | L1 |
| 8 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L1 |
| 9 | L1 | L1 | L2 | L1 | L2 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L2 | L1 | L2 | L1 | L2 | L2 | L2 | L1 |
| 10 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L2 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L2 | L2 |
| 11 | L2 | L1 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | L2 | L1 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 |
| 12 | L1 | L2 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L2 | L2 | L1 | L2 | L1 | L2 | L1 | L1 | L2 | L1 | L1 |
| 13 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L2 | L2 | L2 | L2 | L1 | L2 | L1 | L1 | L2 | L1 | L1 | L1 |
| 14 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L2 |
| 15 | L2 | L2 | L2 | L1 | L2 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | L2 | L1 | L2 | L1 | L1 | L1 | L1 | L2 |
| 16 | L2 | L1 | L1 | L2 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L1 |
| 17 | L1 | L2 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L1 |
| 18 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 |

# All triples in only 18 tests, covering $\binom{20}{3}2^3$ = 9,120 combinations

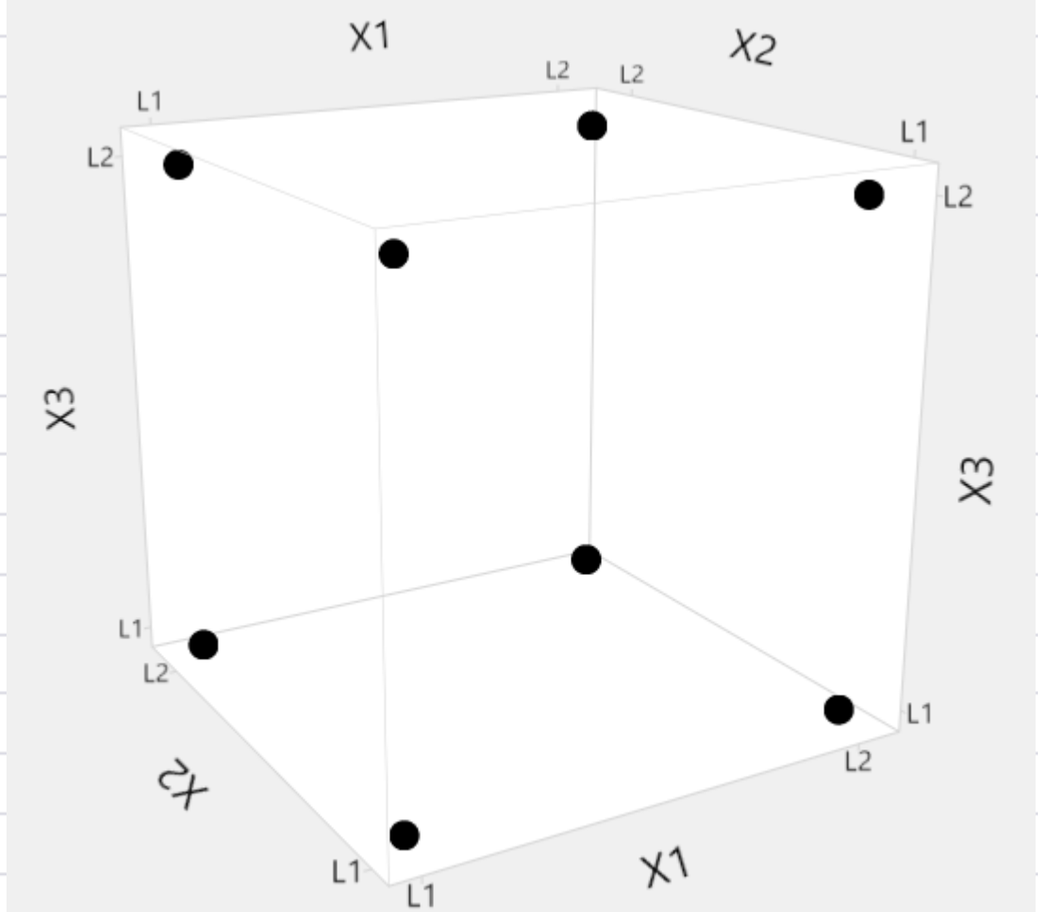
20f 2L Covering Array t3 18 - JMP Pro

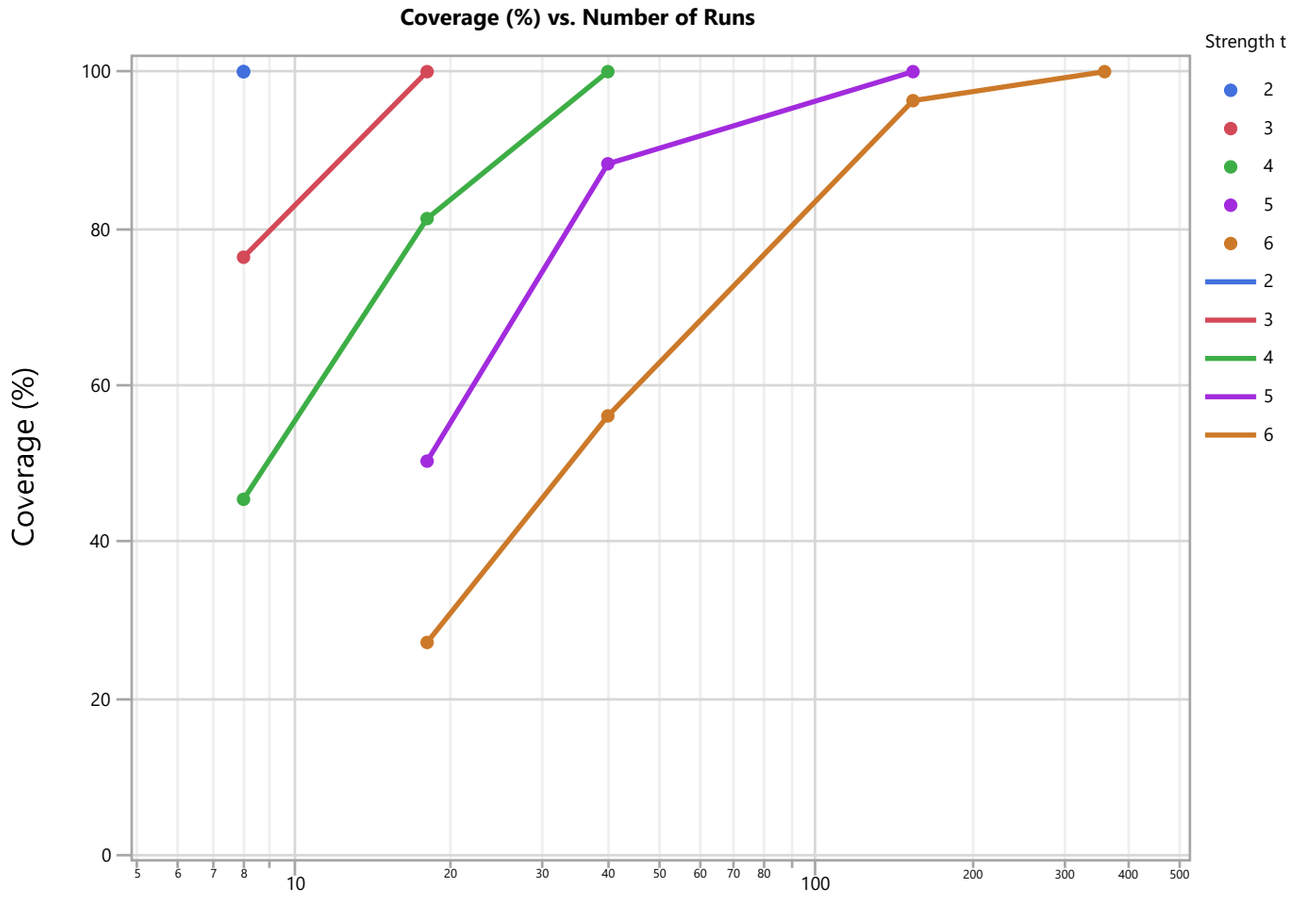
File Edit Tables Rows Cols DOE Analyze Graph Six Sigma Tools Tools Add-Ins View Window Help

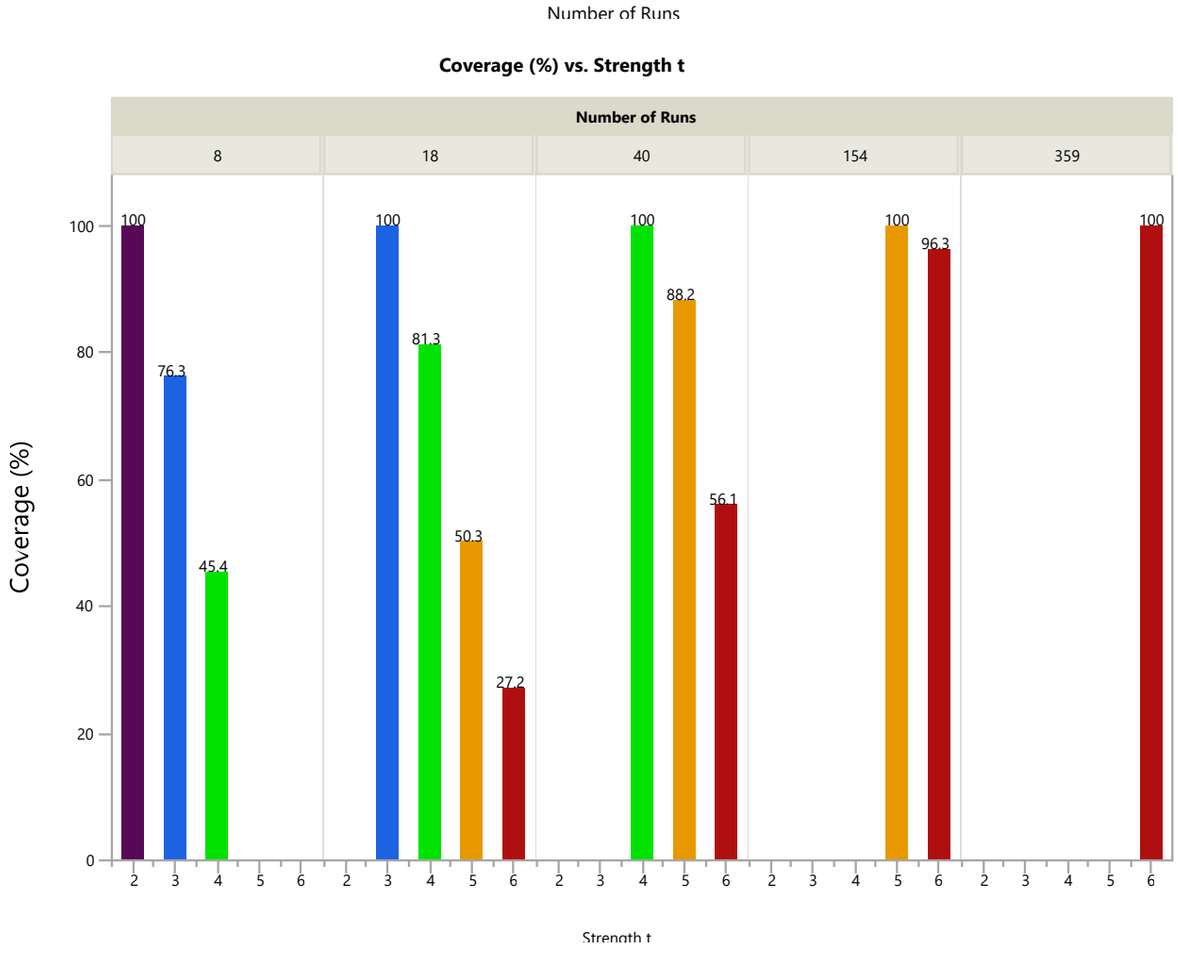
21/0 Cols

18/0

| | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 | X10 | X11 | X12 | X13 | X14 | X15 | X16 | X17 | X18 | X19 | X20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | L1 | L1 | L1 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 |
| 2 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L2 | L2 | L2 | L1 | L1 | L1 |
| 3 | L2 | L2 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L2 | L1 | L1 | L1 | L2 | L2 | L2 | L2 |
| 4 | L2 | L2 | L2 | L2 | L1 | L2 | L1 | L1 | L2 | L2 | L1 | L2 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L2 |
| 5 | L2 | L2 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L2 | L1 |
| 6 | L1 | L2 | L1 | L2 | L1 | L2 | L1 | L2 | L1 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L2 | L2 | L1 | L2 |
| 7 | L2 | L1 | L2 | L2 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L2 | L1 | L2 | L1 | L2 | L1 | L1 |
| 8 | L1 | L2 | L2 | L1 | L2 | L1 | L1 | L2 | L2 | L2 | L1 | L1 | L1 | L1 | L1 | L2 | L1 | L1 | L2 | L1 |
| 9 | L1 | L1 | L2 | L2 | L2 | L2 | L2 | L2 | L1 | L1 | L2 | L1 | L2 | L1 | L2 | L1 | L2 | L2 | L2 | L1 |
| 10 | L1 | L1 | L1 | L2 | L2 | L1 | L1 | L1 | L2 | L1 | L2 | L2 | L2 | L2 | L2 | L2 | L1 | L1 | L2 | L2 |
| 11 | L2 | L1 | L1 | L1 | L1 | L2 | L2 | L2 | L2 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L1 | L2 | L2 | L2 |
| 12 | L1 | L2 | L1 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L2 | L1 | L1 | L1 | L2 | L1 | L1 | L2 | L1 | L1 |
| 13 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L1 | L1 | L2 | L1 | L2 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L1 |
| 14 | L1 | L1 | L2 | L2 | L1 | L1 | L2 | L2 | L2 | L1 | L2 | L2 | L2 | L2 | L1 | L1 | L2 | L1 | L1 | L2 |
| 15 | L2 | L2 | L2 | L2 | L2 | L1 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L1 | L2 | L1 | L1 | L1 | L1 | L2 |
| 16 | L2 | L1 | L1 | L2 | L2 | L1 | L2 | L1 | L1 | L1 | L2 | L1 | L2 | L1 | L1 | L2 | L1 | L1 | L2 | L1 |
| 17 | L1 | L2 | L1 | L2 | L1 | L2 | L1 | L2 | L2 | L1 | L2 | L2 | L2 | L2 | L2 | L1 | L1 | L2 | L1 | L1 |
| 18 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 | L2 |






## Metrics

Number of Runs: 8

| t | Coverage | Diversity |
|---|---|---|
| 2 | 100.00 | 50.00 |
| 3 | 76.32 | 76.32 |
| 4 | 45.43 | 90.87 |

## Metrics

Number of Runs: 18

| t | Coverage | Diversity |
|---|---|---|
| 3 | 100.00 | 44.44 |
| 4 | 81.25 | 72.22 |
| 5 | 50.30 | 89.42 |
| 6 | 27.16 | 96.57 |

## Metrics

Number of Runs: 40

| t | Coverage | Diversity |
|---|---|---|
| 4 | 100.00 | 40.00 |
| 5 | 88.24 | 70.59 |
| 6 | 56.07 | 89.71 |

## Metrics

Number of Runs: 154

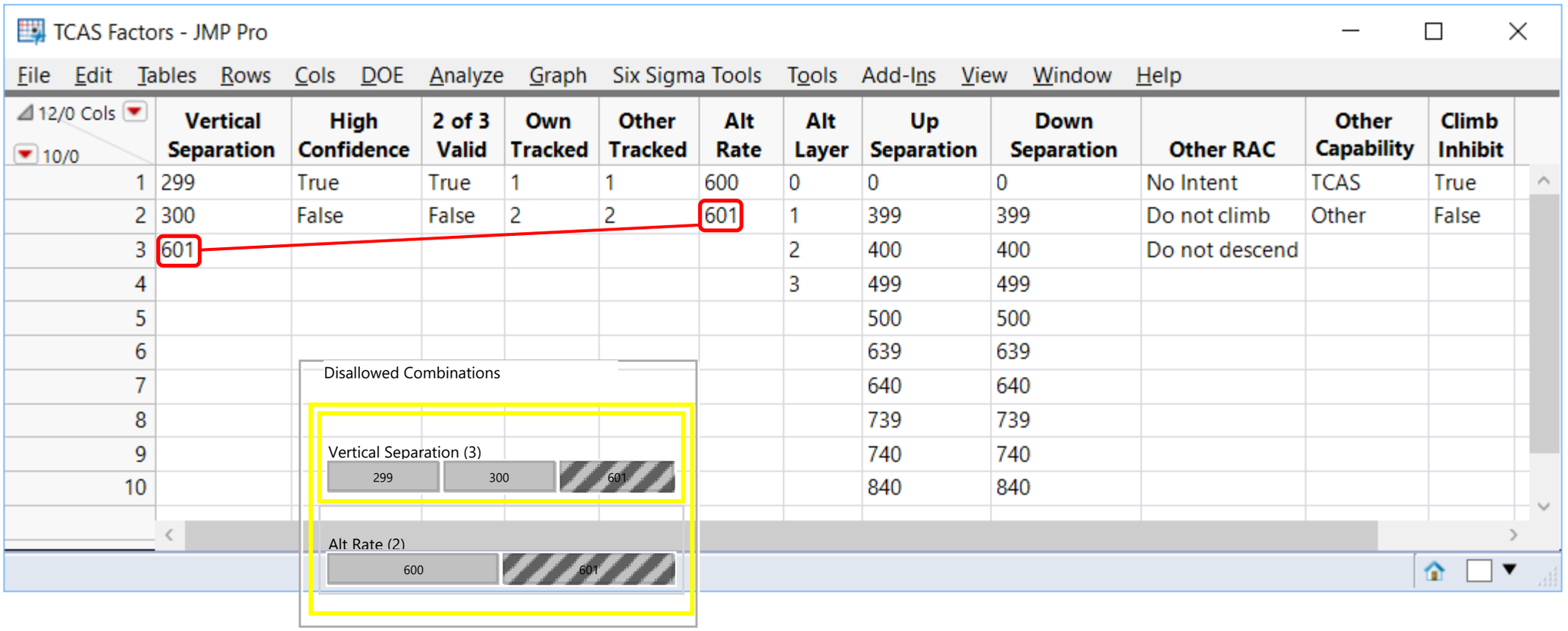
| t | Coverage | Diversity |
|---|---|---|
| 5 | 100.00 | 20.78 |
| 6 | 96.39 | 40.06 |

## Metrics

Number of Runs: 359

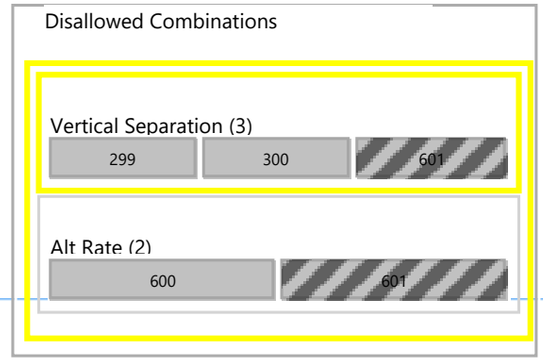
| t | Coverage | Diversity |
|---|---|---|
| 6 | 100.00 | 17.83 |

# Traffic Collision Avoidance System (TCAS)
# $2^7\ 3^2\ 4^1\ 10^2$ = 460,800 possible combinations




# Certain combinations of settings are not allowed. e.g. Vertical Separation = 601 and Alt Rate = 601.

# Sorted Covering Array has 100 rows, first 20 showing Vertical Separation = 601 and Alt Rate ≠ 601




Covering Array - JMP Pro

File Edit Tables Rows Cols DOE Analyze Graph Six Sigma Tools Tools Add-Ins View Window Help

13/2 Cols
100/20

| | Response | Vertical Separation | High Confidence | 2 of 3 Valid | Own Tracked | Other Tracked | Alt Rate | Alt Layer | Up Separation | Down Separation | Other RAC | Other Capability | Climb Inhibit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | • | 601 | True | True | 2 | 2 | 600 | 3 | 499 | 640 | No Intent | TCAS | False |
| 2 | • | 601 | False | True | 2 | 1 | 600 | 3 | 640 | 739 | No Intent | TCAS | True |
| 3 | • | 601 | False | True | 1 | 2 | 600 | 3 | 499 | 399 | No Intent | Other | False |
| 4 | • | 601 | False | True | 2 | 1 | 600 | 1 | 500 | 639 | Do not climb | Other | False |
| 5 | • | 601 | False | False | 1 | 1 | 600 | 0 | 400 | 500 | Do not climb | Other | True |
| 6 | • | 601 | True | False | 1 | 1 | 600 | 3 | 0 | 400 | No Intent | TCAS | True |
| 7 | • | 601 | True | True | 1 | 2 | 600 | 3 | 639 | 740 | Do not descend | TCAS | False |
| 8 | • | 601 | False | False | 1 | 2 | 600 | 0 | 399 | 500 | No Intent | TCAS | False |
| 9 | • | 601 | False | False | 1 | 1 | 600 | 1 | 640 | 399 | Do not descend | TCAS | True |
| 10 | • | 601 | True | False | 2 | 1 | 600 | 0 | 739 | 499 | Do not climb | Other | False |
| 11 | • | 601 | False | False | 2 | 2 | 600 | 2 | 399 | 740 | Do not climb | TCAS | False |
| 12 | • | 601 | False | True | 1 | 2 | 600 | 2 | 739 | 500 | No Intent | Other | False |
| 13 | • | 601 | False | False | 2 | 1 | 600 | 1 | 0 | 639 | Do not climb | TCAS | False |
| 14 | • | 601 | False | True | 1 | 2 | 600 | 2 | 740 | 739 | Do not descend | Other | False |
| 15 | • | 601 | True | True | 1 | 1 | 600 | 1 | 740 | 399 | No Intent | Other | True |
| 16 | • | 601 | False | False | 2 | 2 | 600 | 0 | 840 | 840 | Do not descend | TCAS | False |
| 17 | • | 601 | True | False | 1 | 2 | 600 | 3 | 840 | 639 | Do not climb | TCAS | True |
| 18 | • | 601 | True | False | 1 | 2 | 600 | 1 | 740 | 840 | Do not descend | Other | False |
| 19 | • | 601 | True | False | 1 | 2 | 600 | 3 | 499 | 0 | Do not descend | Other | True |
| 20 | • | 601 | True | False | 1 | 2 | 600 | 1 | 739 | 740 | Do not descend | TCAS | False |
| 21 | • | 300 | False | False | 1 | 2 | 601 | 1 | 739 | 0 | Do not descend | Other | True |
| 22 | • | 300 | False | True | 1 | 2 | 601 | 1 | 740 | 0 | Do not descend | Other | True |
| 23 | • | 300 | True | True | 1 | 2 | 601 | 3 | 500 | 0 | No Intent | TCAS | False |
| 24 | | | | | | | | | | | | | |

# Analysis of TCAS Covering Array
# 2 faults in 1203 run strength 4 design

**Summary**

| | |
|---|---|
| Success Runs | 1201 |
| Failure Runs | 2 |
| Missing | 0 |

**Failure Analysis Details**

**4 Factor Interactions**

| Factors | Failure Levels | Failure Count |
|---|---|---|
| Vertical Separation, High Confidence, Up Separation, Down Separation | 300, True, 399, 500 | 1 |
| Vertical Separation, Own Tracked, Up Separation, Down Separation | 299, 1, 499, 499 | 1 |
| Vertical Separation, Alt Rate, Up Separation, Down Separation | 299, 600, 499, 499 | 1 |
| Vertical Separation, Alt Layer, Up Separation, Down Separation | 299, 1, 499, 499 | 1 |
| Vertical Separation, Alt Layer, Up Separation, Down Separation | 300, 2, 399, 500 | 1 |
| Vertical Separation, Up Separation, Down Separation, Climb Inhibit | 300, 399, 500, False | 1 |
| High Confidence, Alt Rate, Up Separation, Down Separation | True, 601, 399, 500 | 1 |
| High Confidence, Alt Layer, Up Separation, Down Separation | True, 1, 499, 499 | 1 |
| High Confidence, Alt Layer, Up Separation, Down Separation | True, 2, 399, 500 | 1 |
| Own Tracked, Up Separation, Down Separation, Other RAC | 2, 399, 500, No Intent | 1 |
| Alt Layer, Up Separation, Down Separation, Other RAC | 1, 499, 499, Do not descend | 1 |
| Alt Layer, Up Separation, Down Separation, Other RAC | 2, 399, 500, No Intent | 1 |
| Alt Layer, Up Separation, Down Separation, Other Capability | 2, 399, 500, TCAS | 1 |
| Alt Layer, Up Separation, Down Separation, Climb Inhibit | 1, 499, 499, True | 1 |
| Alt Layer, Up Separation, Down Separation, Climb Inhibit | 2, 399, 500, False | 1 |

**These 15 combinations of 4 factors associated with the two detected faults**

# Why Here?

- Attended 5th Cyber Security Workshop and saw little use of statistical methods, particularly design of experiments (DOE) and more specifically covering arrays.
- When analyst for the Army at Edgewood Chemical Biological Center I saw lots of “scenario based” modeling and simulation. Showed that by running a DOE covering the factor space one could include virtually any scenario in the provided analysis – efficiently.
- SAS has been using covering arrays to test its software for more than a decade as a method of quality assurance.
- I am aware of software testers at Eglin AFB and Camp Pendleton using covering arrays.  There are likely others.
- I am hopeful we can excite more Cyber Security Testers to use DOE methods.

# Covering Arrays: Evaluating *t*-Coverage & *t*-Diversity in the presence of disallowed combinations

ITEA 2018

6th Cyber Security Workshop

Joseph Morgan, Ryan Lekivetz, & Tom Donnelly
JMP Division
SAS Institute Incorporated
Cary, North Carolina 27513
Tom.Donnelly@JMP.com

# Covering arrays - Preliminaries

**Definition**

A *covering array* **CA**(N; t, k, v) is an $N \times k$ array such that the $i$-th column contains v distinct symbols. If a **CA**(N; t, k, v) has the property that for any $t$ coordinate projection, all $v^t$ combinations of symbols exist at least once, then it is a ***t***-covering array (or strength t covering array). A ***t***-covering array is optimal if N is minimal for fixed t, k, and v.

**Definition**

The size of a covering array is the *covering array number* **CAN**(t, k, v),

$$\mathbf{CAN}(t, k, v) = \min\{N: \exists\, \mathbf{CA}(N; t, k, v)\}.$$

# Covering arrays - Preliminaries

**Definition**

Consider the covering array $\mathcal{A}$ = **CA**(N; t, k, v). If for any *t* coordinate projection of $\mathcal{A}$ all $v^t$ combinations of symbols exist **exactly λ times** then $\mathcal{A}$ is a ***t***-covering *orthogonal* array of index λ.

***Note*:** Orthogonal arrays are therefore a special type of covering array.

# Covering arrays - Preliminaries

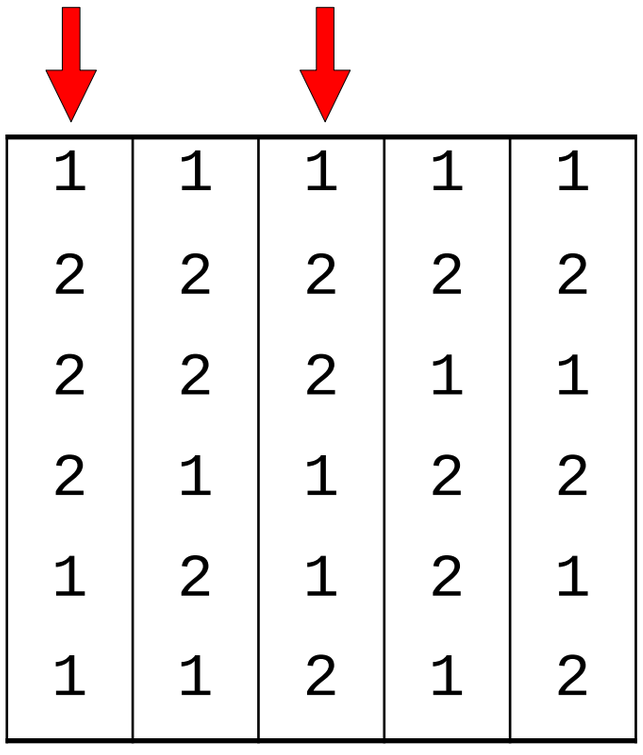
| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

**CA**(12; 3, 5, 2)

# Covering arrays - Preliminaries



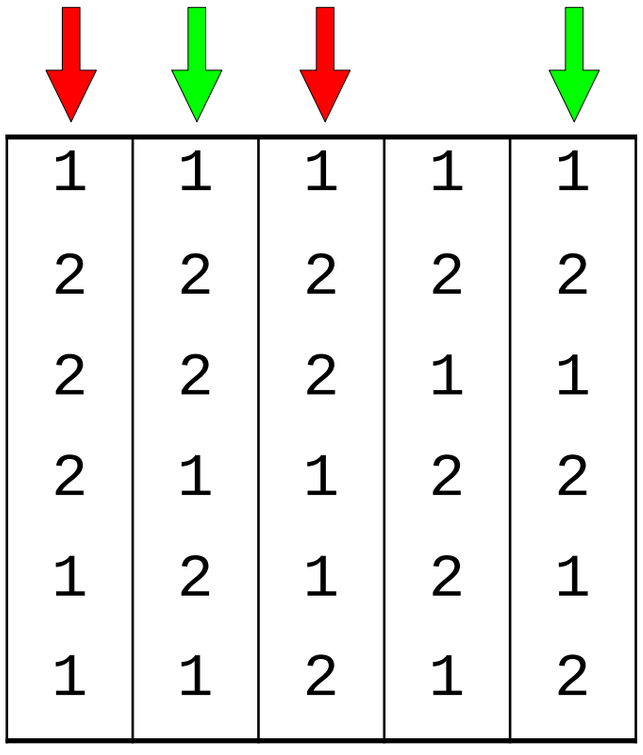
| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

**CA**(12; 3, 5, 2)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

# Covering arrays - Preliminaries



| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

**CA**(12; 3, 5, 2)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

# Covering arrays - Preliminaries

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

**CA**(12; 3, 5, 2)

# Covering arrays - Preliminaries

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

**CA**(12; 3, 5, 2)

# Covering arrays - Preliminaries

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

Optimal[1]

**CA**(6; 2, 5, 2)

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

Not optimal[2]
CAN = 10

**CA**(12; 3, 5, 2)

1 A. Renyi, 1948
2 K. Johnson & R. Entringer, 1989

# Covering arrays - Software & Systems Validation

“*Bugs lurk in corners and congregate at boundaries.*[1]”

[1]B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold, 1983

# Covering arrays - Software & Systems Validation

*“Bugs lurk in corners and congregate at boundaries.[1]”*

*“…choose the type of corner and cover it.”*

[1]B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold, 1983

# Case Study #1

## Air to ground missile system

# Air to ground missile system

Consider a software system controlling the state of an air to ground missile system (Dalal & Mallows[1]).

| | Input | Type |
|---|---|---|
| 1 | Altitude | *continuous* |
| 2 | Attack angle | *continuous* |
| 3 | Bank angle | *continuous* |
| 4 | Speed | *continuous* |
| 5 | Pitch | *continuous* |
| 6 | Roll | *continuous* |
| 7 | Yaw | *continuous* |
| 8 | Ambient Temperature | *continuous* |
| 9 | Pressure | *continuous* |
| 10 | Wind Velocity | *continuous* |

[1]S. R. Dalal & C. L. Mallows, “Factor-covering designs for testing software,” *Technometrics*, 40(3), 1998, 234-243.

# Air to ground missile system

Consider a software system controlling the state of an air to ground missile system (Dalal & Mallows[1]).

| | Input | Type |
|---|---|---|
| 1 | Altitude | *continuous* |
| 2 | Attack angle | *continuous* |
| 3 | Bank angle | *continuous* |
| 4 | Speed | *continuous* |
| 5 | Pitch | *continuous* |
| 6 | Roll | *continuous* |
| 7 | Yaw | *continuous* |
| 8 | Ambient Temperature | *continuous* |
| 9 | Pressure | *continuous* |
| 10 | Wind Velocity | *continuous* |

Challenge:

We are interested in deriving test cases to effectively validate "...response during attack maneuvering."

[1]S. R. Dalal & C. L. Mallows, "Factor-covering designs for testing software," *Technometrics*, 40(3), 1998, 234-243.

# Air to ground missile system

Let the inputs be factors. If the levels are binary then the design specification is:

| | Factor | Levels |
|---|---|---|
| 1 | Altitude | *1, 2* |
| 2 | Attack angle | *1, 2* |
| 3 | Bank angle | *1, 2* |
| 4 | Speed | *1, 2* |
| 5 | Pitch | *1, 2* |
| 6 | Roll | *1, 2* |
| 7 | Yaw | *1, 2* |
| 8 | Ambient Temperature | *1, 2* |
| 9 | Pressure | *1, 2* |
| 10 | Wind Velocity | *1, 2* |

# Air to ground missile system

Let the inputs be factors. If the levels are binary then the design specification is:

| | Factor | Levels |
|---|---|---|
| 1 | Altitude | *1, 2* |
| 2 | Attack angle | *1, 2* |
| 3 | Bank angle | *1, 2* |
| 4 | Speed | *1, 2* |
| 5 | Pitch | *1, 2* |
| 6 | Roll | *1, 2* |
| 7 | Yaw | *1, 2* |
| 8 | Ambient Temperature | *1, 2* |
| 9 | Pressure | *1, 2* |
| 10 | Wind Velocity | *1, 2* |

*Note*: With two levels per factor, the input space is $2^{10} = 1024$ points!

# Air to ground missile system

# Air to ground missile system

“Naive” approach:

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 |
| 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 |

# Air to ground missile system

“Naive” approach:

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 |
| 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 |

# Air to ground missile system

“Naive” approach:

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 |
| 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 |
| | | | 2 | | 1 | | | | |
| | | | 1 | | 2 | | | | |

Missing level combinations!

# Air to ground missile system

Otherwise known as an "All values testing" strategy.

"Naive" approach:

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 2 | 1 | 2 | 2 | 2 | 1 |
| 2 | 2 | 2 | 2 | 1 | 2 | 1 | 1 | 1 | 2 |
| | | | 2 | | 1 | | | | |
| | | | 1 | | 2 | | | | |

Missing level combinations!

# Air to ground missile system

Covering array approach:

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 |
| 2 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 |
| 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 1 |

# Air to ground missile system

Covering array approach:

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 |
| 2 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 |
| 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 1 |

**CA**(6; 2, 10, 2)

Optimal[1]

[1] A. Renyi, 1948

# Covering arrays - Preliminaries

**Definition**[1]

Consider the covering array $\mathcal{A}$ = **CA**(N; t, k, v). Let $n_i$ be the number of distinct *t* tuples for the $i^{th}$ projection, $p_i$ the number of possible *t* tuples *for the* $i^{th}$ projection, *r* the number of rows, and $M = {}^{K}C_t$ the number of *t* column projections that can be obtained from *K* columns.

*Note*: $p_i = \prod_{k=1}^{t} v_{f(k)}$ where $f(k)$ is the $k^{th}$ column of the $i^{th}$ projection.

[1]S. Dalal & C. Mallows, “Factor-covering designs for testing software,” *Technometrics*, 40(3), 1998, 234-243.

# Covering arrays - Preliminaries

**Definition**[1]

Consider the covering array $\mathcal{A}$ = **CA**(N; t, k, v). Let $n_i$ be the number of distinct *t* tuples for the $i^{th}$ projection, $p_i$ the number of possible *t* tuples *for the* $i^{th}$ projection, *r* the number of rows, and $M = {}_K C_t$ the number of *t* column projections that can be obtained from *K* columns.

$$t_{\mathcal{A}}\text{-Coverage} = \frac{\mathbf{1}}{\boldsymbol{M}} \sum_{i=1}^{M} n_i / p_i .$$

*Note*: $p_i = \prod_{k=1}^{t} v_{f(k)}$ where $f(k)$ is the $k^{th}$ column of the $i^{th}$ projection.

[1]S. Dalal & C. Mallows, “Factor-covering designs for testing software,” *Technometrics*, 40(3), 1998, 234-243.

# Covering arrays - Preliminaries

**Definition**[1]

Consider the covering array $\mathcal{A}$ = **CA**(N; t, k, v). Let $n_i$ be the number of distinct *t* tuples for the $i^{th}$ projection, $p_i$ the number of possible *t* tuples *for the* $i^{th}$ projection, *r* the number of rows, and $M = {}_K C_t$ the number of *t* column projections that can be obtained from *K* columns.

$$t_{\mathcal{A}}\text{-Coverage} = \frac{\mathbf{1}}{\boldsymbol{M}} \sum_{i=1}^{M} n_i / p_i .$$

$$t_{\mathcal{A}}\text{-Diversity} = \frac{\mathbf{1}}{\boldsymbol{M}} \sum_{i=1}^{M} n_i / r .$$

*Note*: $p_i = \prod_{k=1}^{t} v_{f(k)}$ where $f(k)$ is the $k^{th}$ column of the $i^{th}$ projection.

[1]S. Dalal & C. Mallows, “Factor-covering designs for testing software,” *Technometrics*, 40(3), 1998, 234-243.

# Covering arrays - Preliminaries

***$t_A$*-Coverage:**

The ratio of the number of distinct *t* tuples, to the number of **possible** *t* tuples, averaged over all *t* column projections (1≤t≤K).

***$t_A$*-Diversity**:

The ratio of the number of distinct *t* tuples, to the **total** number of *t* tuples, averaged over all *t* column projections (1≤t≤K).

# Covering arrays - Preliminaries

*t*-Coverage =

*t*-Diversity =

# Covering arrays - Preliminaries

$t$-Coverage = $\frac{1}{M}\sum_{i=1}^{M} n_i/p_i$

$t$-Diversity = $\frac{1}{M}\sum_{i=1}^{M} n_i/r$

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

# Covering arrays - Preliminaries

$t$-Coverage = $\frac{1}{M}\sum_{i=1}^{M} n_i/p_i$

$t$-Diversity = $\frac{1}{M}\sum_{i=1}^{M} n_i/r$

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 1 | 2 | 1 | 2 | 1 |
| 1 | 1 | 2 | 1 | 2 |

**CA**(6; 2, 5, 2)

| *t* | Coverage | Diversity |
|---|---|---|
| 2 | 100.0 | 66.7 |
| 3 | 70.0 | 93.3 |
| 4 | 37.5 | 100.0 |

# Covering arrays - Preliminaries

$t$-Coverage = $\frac{1}{M}\sum_{i=1}^{M} n_i/p_i$

$t$-Diversity = $\frac{1}{M}\sum_{i=1}^{M} n_i/r$

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

**CA**(12; 3, 5, 2)

$t$-Coverage = $\frac{1}{M}\sum_{i=1}^{M} n_i/p_i$

$t$-Diversity = $\frac{1}{M}\sum_{i=1}^{M} n_i/r$

# Covering arrays - Preliminaries

| $t$ | Coverage | Diversity |
|---|---|---|
| 3 | 100.0 | 66.7 |
| 4 | 70.0 | 93.3 |
| 5 | 37.5 | 100.0 |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 1 | 1 |
| 1 | 2 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 1 | 2 | 2 | 1 |
| 2 | 2 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 1 | 1 | 1 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 |
| 2 | 1 | 2 | 1 | 2 |
| 1 | 2 | 2 | 2 | 1 |

**CA**(12; 3, 5, 2)

# Covering arrays - Preliminaries

*t*-Coverage =

*t*-Diversity =

# Covering arrays - Preliminaries

$t$-Coverage = $\frac{1}{M}\sum_{i=1}^{M} n_i/p_i$

$t$-Diversity = $\frac{1}{M}\sum_{i=1}^{M} n_i/r$

Air to ground missile system, **CA**(6; 2, 10, 2):

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 |
| 2 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 |
| 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 1 |

# Covering arrays - Preliminaries

$t$-Coverage = $\frac{1}{M}\sum_{i=1}^{M} n_i/p_i$

$t$-Diversity = $\frac{1}{M}\sum_{i=1}^{M} n_i/r$

Air to ground missile system, **CA**(6; 2, 10, 2):

| ***t*** | **Coverage** | **Diversity** |
|---|---|---|
| 2 | 100.0 | 66.7 |
| 3 | 68.8 | 91.7 |
| 4 | 37.1 | 98.8 |

| Altitude | Attack Angle | Bank Angle | Speed | Pitch | Roll | Yaw | Ambient Temp | Pressure | Wind Velocity |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 |
| 2 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 1 | 2 |
| 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 1 |

# Covering arrays - Preliminaries

**Definition**

A *mixed level covering array* **MCA**(N; t, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$)) is an $N \times k$ array such that the $i$-th column contains $v_i$ distinct symbols. If for any $t$ coordinate projection, all $\prod v_i$ combinations of symbols exist, then it is a ***t***-covering array and is optimal if N is minimal for fixed t, k, and ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$).

# Covering arrays - Preliminaries

**Definition**

A *mixed level covering array* **MCA**(N; t, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$)) is an $N \times k$ array such that the $i$-th column contains $v_i$ distinct symbols. If for any $t$ coordinate projection, all $\prod v_i$ combinations of symbols exist, then it is a ***t***-covering array and is optimal if N is minimal for fixed t, k, and ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$).

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 |
| 1 | 3 | 1 | 1 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 1 | 1 | 2 |
| 2 | 3 | 2 | 1 | 1 |
| 3 | 1 | 1 | 1 | 2 |
| 3 | 2 | 2 | 1 | 1 |
| 3 | 3 | 2 | 2 | 2 |

# Covering arrays - Preliminaries

**Definition**

A *mixed level covering array* **MCA**(N; t, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$)) is an $N \times k$ array such that the $i$-th column contains $v_i$ distinct symbols. If for any $t$ coordinate projection, all $\prod v_i$ combinations of symbols exist, then it is a ***t***-covering array and is optimal if N is minimal for fixed t, k, and ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$).

An optimal **MCA**(9; 2, ($3^2 \cdot 2^3$)).

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 |
| 1 | 3 | 1 | 1 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 1 | 1 | 2 |
| 2 | 3 | 2 | 1 | 1 |
| 3 | 1 | 1 | 1 | 2 |
| 3 | 2 | 2 | 1 | 1 |
| 3 | 3 | 2 | 2 | 2 |

# Covering arrays - Preliminaries

**Definition**

A *mixed level covering array* **MCA**(N; t, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$)) is an $N \times k$ array such that the $i$-th column contains $v_i$ distinct symbols. If for any $t$ coordinate projection, all $\prod v_i$ combinations of symbols exist, then it is a ***t***-covering array and is optimal if N is minimal for fixed t, k, and ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$).

| ***t*** | **Coverage** | **Diversity** |
|---|---|---|
| 2 | 100.0 | 63.3 |
| 3 | 65.0 | 93.3 |
| 4 | 30.0 | 100.0 |

$t$-Coverage = $\frac{\mathbf{1}}{\boldsymbol{M}} \sum_{i=1}^{M} n_i / p_i .$

$t$-Diversity = $\frac{\mathbf{1}}{\boldsymbol{M}} \sum_{i=1}^{M} n_i / r .$

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 2 |
| 1 | 2 | 2 | 2 | 1 |
| 1 | 3 | 1 | 1 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 1 | 1 | 2 |
| 2 | 3 | 2 | 1 | 1 |
| 3 | 1 | 1 | 1 | 2 |
| 3 | 2 | 2 | 1 | 1 |
| 3 | 3 | 2 | 2 | 2 |

**MCA**(9; 2, ($3^2 \cdot 2^3$))

# Covering arrays - Preliminaries

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Covering arrays - Preliminaries

In practice, systems and software validation problems come with constraints. Consider a GUI that contains a variety of controls, such as checkboxes, combo boxes, etc., where some setting of a particular control precludes settings of some other control.

In discussing testing configurable systems, Myra Cohen[1] makes the point:

> "*Constraints may arise due to any number of reasons, for example, inconsistencies between certain hardware components, limitations due to available memory and software size, or simply marketing decisions.*"

Constraints are sometimes referred to as *disallowed combinations* or *forbidden interactions*.

[1]M. Cohen, M. Dwyer, J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," IEEE TSE, 2008, 633-650.

# Covering arrays - Preliminaries

MCA's are not sufficient, we need another generalization!

# Constrained covering arrays

**Definition**[1]

A *constrained covering array* **CCA**(N; t, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$), $\phi$) is an $N \times k$ array such that the *i*-th column contains $v_i$ distinct symbols and $\phi$ is a set of *p*-tuples ($2 \leq p \leq k$) such that each tuple is a set of two or more column/value pairs identifying a disallowed combination. If for any *t* coordinate projection, all **possible** combinations of symbols exist, then it is a ***t***-covering array and is optimal if N is minimal for fixed t, k, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$), and $\phi$.

*Note*: Let us say that the symbol set for columns $c_1$ and $c_2$ is {1, 2} and the symbol combination (2, 1) is not allowed then $\phi = \{(c_1,2), (c_2,2)\}$.

[1]J. Morgan, “Combinatorial Testing: An approach to systems and software testing based on covering arrays,” in *Analytic Methods in Systems and Software Testing,* eds., F. Ruggeri, R. Kennett, & F. Faltin, Wiley, 2018.

# Constrained covering arrays

Consider a **CCA**(9; 2, ($3^2 \cdot 2^3$),ϕ ) where ϕ ={{($c_1$,1),($c_3$,1)}}.

It is usually convenient to express ϕ more compactly. Cohen[1], suggests a shorthand exponent notation, $p^k$, to indicate k *p*-tuples. In this case, ϕ = {$2^1$}.

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Constrained covering arrays

Consider a **CCA**(9; 2, ($3^2 \cdot 2^3$),ϕ ) where ϕ ={{($c_1$,1),($c_3$,1)}}.

It is usually convenient to express ϕ more compactly. Cohen[1], suggests a shorthand exponent notation, $p^k$, to indicate k *p*-tuples. In this case, ϕ = {$2^1$}.

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 1 |
| 1 | 2 | 2 | 1 | 2 |
| 1 | 3 | 2 | 2 | 2 |
| 2 | 1 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 3 | 2 | 1 | 1 |
| 3 | 1 | 2 | 1 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 3 | 3 | 1 | 1 | 2 |

**CCA**(9; 2, ($3^2 \cdot 2^3$), ϕ ={$2^1$})

Optimal

Exponent notation

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Case Study #2

## Mobile Phone Configuration

# Constrained covering arrays

Consider a mobile phone configuration problem (Cohen et al.[1]):

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Covering arrays - Phone constraints[1]

I. Graphical email viewer requires a color display.

II. 2 megapixel camera requires a color display.

III. Graphical email viewer is not supported with 2 megapixel cameras.

IV. 8 million color display does not support a 2 megapixel camera.

V. Video camera requires a camera and a color display.

VI. Video ringtones cannot occur without a video camera.

VII. 16 million colors, text, and 2 megapixel camera not supported.

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Covering arrays - Phone constraints[1]

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

I. Graphical email viewer requires a color display.

II. 2 megapixel camera requires a color display.

III. Graphical email viewer is not supported with 2 megapixel cameras.

IV. 8 million color display does not support a 2 megapixel camera.

V. Video camera requires a camera and a color display.

VI. Video ringtones cannot occur without a video camera.

VII. 16 million colors, text, and 2 megapixel camera not supported.

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Covering arrays - Disallowed combinations

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

I. **Display**: B&W, **Email Viewer**: Graphical.

II. **Display**: B&W, **Camera**: 2 megapixel.

III. **Email Viewer**: Graphical, **Camera**: 2 megapixel.

IV. **Display**: 8 million colors, **Camera**: 2 megapixel.

V. (**Display**: B&W, **Video Camera**: Y) & (**Camera**: None, **Video Camera**: Y).

VI. **Video Camera**: N, **Video Ringtones**: N.

VII. **Display**: 16 million colors, **Email Viewer**: Text, **Camera**: 2 megapixel.

[1]M. Cohen, M. Dwyer, J. Shi, “Interaction testing of highly-configurable systems in the presence of constraints,” IEEE TSE, 2008, 633-650.

# Constrained covering arrays - Phone example

**CCA**(10; 2, $(3^3 \cdot 2^2)$, $\phi = \{2^7 \cdot 3^1\}$)

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**$(10; 2, (3^3 \cdot 2^2), \phi = \{2^7 \cdot 3^1\})$

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**$(10; 2, (3^3 \cdot 2^2), \phi = \{2^7 \cdot 3^1\})$

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

**Display:** B&W,

**Email Viewer:** Graphical

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**(10; 2, ($3^3 \cdot 2^2$), $\phi = \{2^7 \cdot 3^1\}$)

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

**Display:** B&W, **Camera:** 2MP

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**(10; 2, $(3^3 \cdot 2^2)$, $\phi = \{2^7 \cdot 3^1\}$)

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

**Email Viewer:** Graphical,

**Camera:** 2MP

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**$(10; 2, (3^3 \cdot 2^2), \phi = \{2^7 \cdot 3^1\})$

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

**Display:** 8 mill, **Camera:** 2MP

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**$(10; 2, (3^3 \cdot 2^2), \phi = \{2^7 \cdot 3^1\})$

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

(**Display:** B&W, **Video Camera:** Y) &

(**Camera:** None, **Video Camera:** Y)

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**(10; 2, $(3^3 \cdot 2^2)$, $\phi = \{2^7 \cdot 3^1\}$)

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

**Video Camera:** N,

**Video Ringtones:** N

# Constrained covering arrays - Phone example

| Component | Setting |
|---|---|
| Display | 16 million colors<br>8 million colors<br>Black & White |
| Email viewer | Graphical<br>Text None |
| Camera | 2 Megapixel<br>1 Megapixel<br>None |
| Video Camera | Yes<br>No |
| Video Ringtones | Yes<br>No |

**CCA**$(10; 2, (3^3 \cdot 2^2), \phi = \{2^7 \cdot 3^1\})$

| Display | Email Viewer | Camera | Video Camera | Video Ringtones |
|---|---|---|---|---|
| 16 mill | Graphical | None | N | N |
| 16 mill | Text | 2MP | N | N |
| 16 mill | None | 2MP | Y | Y |
| 16 mill | Graphical | 1MP | Y | Y |
| 8 mill | Graphical | 1MP | N | N |
| 8 mill | Text | 1MP | Y | Y |
| 8 mill | Text | None | N | N |
| 8 mill | None | 1MP | Y | N |
| B&W | Text | 1MP | N | N |
| B&W | None | None | N | N |

Disallowed combination:

**Display:** 16 mill,

**Email Viewer:** Text,

**Camera:** 2MP

# Constrained covering arrays - Examples[1]

1. **Bugzilla 2.22.2:**
   1.1. Bugzilla is an open source defect tracking system from Mozilla.
   1.2. From the *Administering Bugzilla, Using Bugzilla*, and *Customizing Bugzilla* chapters, 5 constraints involving 11 options were found.
   1.3. We need a **CCA**(N; t, $(2^{49} \cdot 3^{1} \cdot 4^{2}), \phi = \{2^{4} \cdot 3^{1}\}$) design. The unconstrained configuration space is $2.7 \times 10^{16}$.

   *e.g.*: When "Mail Transfer Agent" is "Postfix", "sendmailnow" option must be on.

2. **GCC 4.1 - Optimizer only:**
   1. GCC is a compiler infrastructure with support for multiple languages (e.g. C, C++, Ada) and over 30 different target machine architectures.
   2. From the documentation, 40 constraints involving 35 options were found.
   3. We need a **CCA**(N; t, $(2^{189} \cdot 3^{10}), \phi = \{2^{37} \cdot 3^{3}\}$) design. The unconstrained configuration space is $4.6 \times 10^{61}$.

[1]M. Cohen, M. Dwyer, J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," IEEE TSE, 2008, 633-650.

# Constrained covering arrays - Examples[1]

1. **Bugzilla 2.22.2:** For $t$=2, C-CAN = 16

   1.1. Bugzilla is an open source defect tracking system from Mozilla.

   1.2. From the *Administering Bugzilla, Using Bugzilla*, and *Customizing Bugzilla* chapters, 5 constraints involving 11 options were found.

   1.3. We need a **CCA**(N; t, $(2^{49} \cdot 3^{1} \cdot 4^{2}), \phi = \{2^{4} \cdot 3^{1}\}$) design. The unconstrained configuration space is $2.7 \times 10^{16}$.

   *e.g.*: When "Mail Transfer Agent" is "Postfix", "sendmailnow" option must be on.

2. **GCC 4.1 - Optimizer only:** For $t$=2, C-CAN ≤ 20

   1. GCC is a compiler infrastructure with support for multiple languages (e.g. C, C++, Ada) and over 30 different target machine architectures.

   2. From the documentation, 40 constraints involving 35 options were found.

   3. We need a **CCA**(N; t, $(2^{189} \cdot 3^{10}), \phi = \{2^{37} \cdot 3^{3}\}$) design. The unconstrained configuration space is $4.6 \times 10^{61}$.

[1]M. Cohen, M. Dwyer, J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," IEEE

# Constrained covering arrays

# Constrained covering arrays

As defined CCA's are still not sufficient!

# Constrained covering arrays

Consider a **CCA**(9; 2, $(3^2 \cdot 2^3)$,$\phi$ ) where $\phi$ ={{($c_1$,1),($c_3$,1)}, {($c_1$,1),($c_3$,2)}}.

# Constrained covering arrays

Consider a **CCA**(9; 2, $(3^2 \cdot 2^3)$,$\phi$ ) where $\phi$ ={{($c_1$,1),($c_3$,1)}, {($c_1$,1),($c_3$,2)}}.

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | . | 1 | 1 |
| 1 | 2 | . | 2 | 2 |
| 1 | 3 | . | 2 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 3 | 2 | 1 | 2 |
| 3 | 1 | 2 | 2 | 2 |
| 3 | 2 | 1 | 1 | 1 |
| 3 | 3 | 1 | 2 | 2 |

**CCA**(9; 2, $(3^2 \cdot 2^3)$, $\phi$ ={$2^2$})

# Constrained covering arrays

Consider a **CCA**(9; 2, $(3^2 \cdot 2^3)$,$\phi$ ) where $\phi$ ={{($c_1$,1),($c_3$,1)}, {($c_1$,1),($c_3$,2)}}.

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | . | 1 | 1 |
| 1 | 2 | . | 2 | 2 |
| 1 | 3 | . | 2 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 3 | 2 | 1 | 2 |
| 3 | 1 | 2 | 2 | 2 |
| 3 | 2 | 1 | 1 | 1 |
| 3 | 3 | 1 | 2 | 2 |

**CCA**(9; 2, $(3^2 \cdot 2^3)$, $\phi$ ={$2^2$})

Optimal

# Constrained covering arrays

Consider a **CCA**(9; 2, $(3^2 \cdot 2^3)$,$\phi$ ) where $\phi$ ={{($c_1$,1),($c_3$,1)}, {($c_1$,1),($c_3$,2)}}.

When $c_1$ is set to 1 there are no allowable settings for column $c_3$. The symbol "." is used to indicate an unassigned cell.

***Note***: We refer to such covering arrays as *unsatisfiable* constrained covering arrays.

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | . | 1 | 1 |
| 1 | 2 | . | 2 | 2 |
| 1 | 3 | . | 2 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 3 | 2 | 1 | 2 |
| 3 | 1 | 2 | 2 | 2 |
| 3 | 2 | 1 | 1 | 1 |
| 3 | 3 | 1 | 2 | 2 |

**CCA**(9; 2, $(3^2 \cdot 2^3)$, $\phi$ ={$2^2$})

Optimal

# Constrained covering arrays

**Definition**[1]

Consider $\mathcal{A}$ = **CCA**(N; t, ($v_1 \cdot v_2 \cdot \ldots \cdot v_k$), ϕ ). For the $i^{th}$ projection, let $n_i$ be the number of distinct $t$ tuples, $p_i$ the number of possible $t$ tuples, $a_i$ the number of invalid $t$ tuples, $M' = {}_K C_t - m$, ($m$ the number of projections where there are no valid $t$ tuples), and $r_i = r - q_i$, ($q_i$ the number of rows with missing values).

Adjusted $t_{\mathcal{A}}$-Coverage = $\frac{1}{M'} \sum_{i=1}^{M'} n_i / (p_i - a_i)$

Adjusted $t_{\mathcal{A}}$-Diversity = $\frac{1}{M'} \sum_{i=1}^{M'} n_i / r_i$

[1]J. Morgan, “Combinatorial Testing: An approach to systems and software testing based on covering arrays,” in *Analytic Methods in Systems and Software Testing,* eds., F. Ruggeri, R. Kennett, & F. Faltin, Wiley, 2018.

# Constrained covering arrays

**Adjusted $t_{\mathcal{A}}$-Coverage:**

The ratio of the number of distinct *t* tuples, to the number of **adjusted possible** *t* tuples, averaged over all *t* column projections ($1 \leq t \leq K$) that contain valid tuples.

**Adjusted $t_{\mathcal{A}}$-Diversity**:

The ratio of the number of distinct *t* tuples, to the **adjusted total** number of *t* tuples, averaged over all *t* column projections ($1 \leq t \leq K$) that contain valid tuples.

# Constrained covering arrays

Adjusted $t$-Coverage = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/(p_i - a_i)$

Adjusted $t$-Diversity = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/r_i$

Adjusted *t*-Coverage = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/(p_i - a_i)$

Adjusted *t*-Diversity = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/r_i$

# Constrained covering arrays

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 1 |
| 1 | 2 | 2 | 1 | 2 |
| 1 | 3 | 2 | 2 | 2 |
| 2 | 1 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 3 | 2 | 1 | 1 |
| 3 | 1 | 2 | 1 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 3 | 3 | 1 | 1 | 2 |

**CCA**(9; 2, ($3^2 \cdot 2^3$), $\phi = \{2^1\}$)

# Constrained covering arrays

Adjusted *t*-Coverage = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/(p_i - a_i)$

Adjusted *t*-Diversity = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/r_i$

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 1 |
| 1 | 2 | 2 | 1 | 2 |
| 1 | 3 | 2 | 2 | 2 |
| 2 | 1 | 1 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 |
| 2 | 3 | 2 | 1 | 1 |
| 3 | 1 | 2 | 1 | 1 |
| 3 | 2 | 1 | 2 | 1 |
| 3 | 3 | 1 | 1 | 2 |

**CCA**(9; 2, $(3^2 \cdot 2^3)$, $\phi = \{2^1\}$)

| *t* | Adjusted Coverage | Adjusted Diversity |
|---|---|---|
| 2 | 100.0 | 62.2 |
| 3 | 68.3 | 92.2 |
| 4 | 33.5 | 100.0 |

# Constrained covering arrays

Adjusted $t$-Coverage = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/(p_i - a_i)$

Adjusted $t$-Diversity = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/r_i$

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | . | 1 | 1 |
| 1 | 2 | . | 2 | 2 |
| 1 | 3 | . | 2 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 3 | 2 | 1 | 2 |
| 3 | 1 | 2 | 2 | 2 |
| 3 | 2 | 1 | 1 | 1 |
| 3 | 3 | 1 | 2 | 2 |

**CCA**$(9; 2, (3^2 \cdot 2^3), \phi = \{2^2\})$

Adjusted *t*-Coverage = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/(p_i - a_i)$

Adjusted *t*-Diversity = $\frac{1}{M'}\sum_{i=1}^{M'} n_i/r_i$

# Constrained covering arrays

| *t* | Adjusted Coverage | Adjusted Diversity |
|---|---|---|
| 2 | 100.0 | 71.1 |
| 3 | 59.6 | 96.1 |
| 4 | 27.5 | 100.0 |

| c | c | c | c | c |
|---|---|---|---|---|
| 1 | 1 | . | 1 | 1 |
| 1 | 2 | . | 2 | 2 |
| 1 | 3 | . | 2 | 1 |
| 2 | 1 | 1 | 2 | 1 |
| 2 | 2 | 2 | 1 | 1 |
| 2 | 3 | 2 | 1 | 2 |
| 3 | 1 | 2 | 2 | 2 |
| 3 | 2 | 1 | 1 | 1 |
| 3 | 3 | 1 | 2 | 2 |

**CCA**$(9; 2, (3^2 \cdot 2^3), \phi = \{2^2\})$

# Covering arrays - Software Validation

# Covering arrays - Software Validation

Question: Are covering arrays effective tools for deriving test suites to validate software systems?

# Covering arrays - Software Validation

Question: Are covering arrays effective tools for deriving test suites to validate software systems?

Answer: Kuhn et al.[1] examined several classes of software systems and discovered the following:

| | % faults detected | | | |
|---|---|---|---|---|
| | t=2 | t=3 | t=4 | t=5 |
| **Medical device software** | 95 | 99 | 100 | 100 |
| **Browser application** | 70 | 90 | 95 | 96 |
| **Server software** | 75 | 95 | 96 | 99 |
| **Network security** | 62 | 89 | 99 | 100 |
| **TCAS** | 54 | 74 | 89 | 100 |

Only 3600 test cases, < 1% of exhaustive testing!

[1]D. Kuhn & D. Wallace, & A. Gallo, “Software fault interactions and implications for software testing,” IEEE Trans. SE, v30(6) (2004), 418-421.

# Covering arrays - Software Validation

**Takeaway**:

- Given a software system with *k* inputs, a strength *t* covering array may be used to generate a test suite in which all *t*-way interactions are covered by at least one test case. If *t* << *k* then the cost savings is dramatic when compared to exhaustive testing.
- Covering arrays are an effective and efficient tool for deriving test suites to validate software systems.

# Covering arrays - Software Validation

**Takeaway**:

- Given a software system with *k* inputs, a strength *t* covering array may be used to generate a test suite in which all *t*-way interactions are covered by at least one test case. If *t* << *k* then the cost savings is dramatic when compared to exhaustive testing.
- Covering arrays are an effective and efficient tool for deriving test suites to validate software systems.

*Pseudo-exhaustive* testing[1]

[1]D. Kuhn & V. Okum, “Pseudo-exhaustive testing for software,” Proc. 30th IEEE/NASA Software Engineering Workshop, (2006).

# References


1. R. Brownlie, J. Prowse, & M. S. Padke. Robust testing of AT&T PMX/StarMAIL using OATS. *AT&T Technical Journal*, 71(3), 1992, 41-47.
2. B. Beizer, *Software Testing Techniques*, Van Nostrand Reinhold, 1983.
3. D. M. Cohen, S. R. Dalal, M. L. Fredman, & G. C. Patton, "The AETG System: An approach to testing based on Combinatorial Design," *IEEE TSE*, 23(7), 1997, 437-444.
4. D. M. Cohen, S. R. Dalal, J. Parelius, & G. C. Patton, "The combinatorial design approach to automatic test generation," *IEEE Software*, 13(5), 1996, pp 83-88.
5. M. B. Cohen, M. B. Dwyer,& J. Shi, "Constructing interaction test suites for highly configurable systems in the presence of constraints: A greedy approach," *IEEE TSE*, 34(5), 2008, 633-650.
6. S. R. Dalal, A. J. Karunanithi, J. M. Leaton, G. C. Patton, & B. M. Horowitz, "Model-based testing in practice," *Proceedings of the 21st ICSE*, New York, 1999, 285-294.
7. S. R. Dalal & C. L. Mallows, "Factor-covering designs for testing software," *Technometrics*, 40(3), 1998, 234-243.
8. Federal Aviation Administration, "Introduction to TCAS II v7.1," Tech. Rep. HQ-111358, 2011.
9. I. S. Dunietz, W. K. Ehrlich, B. D. Szablak, C. L. Mallows, & A. Iannino, "Applying design of experiments to software testing," *Proceedings of the 19th ICSE*, New York, 1997, 205-215.
10. A. Hartman & L. Raskin, "Problems and algorithms for covering arrays," *Discrete Math*, 284(1–3), 2004, 149–156.
11. K. A. Johnson, & R. Entringer, "Largest induced subgraphs of the n-cube that contain no 4-cycles," *Journal of Combinatorial Theory, Series B*, 46(3), 1989, 346-355.
12. G. Katona, "Two applications (for search theory and truth functions) of Sperner type theorems," *Periodica Mathematica Hungarica*, 3(1-2), 1973, 19-26.

# References


14. D. Kleitman & J. Spencer, “Families of k-independent sets,” *Discrete Mathematics*, 6(3), 1973, 255-262.
15. D. Kuhn, D. R. Wallace, & A. M. Gallo, “Software Fault Interactions & Implications for Software Testing,” *IEEE TSE*, 30(6), 2004, 418-421.
16. R. Mandl, “Orthogonal latin squares: an application of experiment design to compiler testing,” *Communications of the ACM*, 28(10), 1985, 1054-1058.
17. J. A. Morgan, G. J. Knafl, & W. E. Wong, “Predicting Fault Detection Effectiveness,” *Proceedings of the 4th ISSM*, 1997, 82-90.
18. J. A. Morgan, “A survey of strength 3 MCA algorithms,” *SAS Technical Report*, 2014.
19. J. A. Morgan, “Combinatorial Testing: An approach to systems and software testing based on covering arrays,” in *Analytic Methods in Systems and Software Testing,* eds., F. Ruggeri, R. Kennett, & F. Faltin, Wiley, 2018
20. L. Moura, J. Stardom, B. Stevens, A. Williams, “Covering arrays with mixed alphabet sizes,” *Journal of Combinatorial Design*, 11(6), 2003, 413-432.
21. G. J. Myers, The Art of Software Testing, Wiley, 1979.
22. A. Renyi, Foundations of Probability, Wiley, 1971.

# Thank You