Towards Accessibility of Covering Arrays for Practitioners of Combinatorial Testing

Ulrike Grömping

31 March 2025,

IWCT14, Naples

BHT
Berliner Hochschule für Technik

Contents

1 Introduction

2 Current status

3 Implementation of mathematical constructions

4 Future of Colbourn tables and corresponding CAs

5 Call for input

6 References

My background

Statistician, R software, expertise on **O**rthogonal **A**rrays (OAs), until recently **U**n**I**nitiated **P**ractitioner (UIP) regarding **C**overing **A**rrays (CAs)

Sabbatical project: provide R package for CAs with basic tools around them, **accessible for UIPs**

- **focused on mathematical constructions**
- potentially with post-optimization (like simulated annealing or tabu search)
- **with access to catalogued CAs**
- with evaluation of coverage (for small and perhaps moderately-sized CAs, otherwise only via sampling subsets of columns)
- ***with API access to external search tools like ACTS*** ([17])***, CAgen*** ([5]) ***or CTwedge*** ([8])
- initially index $\lambda = 1$ only, may change later
- later perhaps also with analysis facilities
- later perhaps also with LAs and/or DAs

Goal of this talk

- share my fresh eyes look at the status regarding CA availability for practitioners
- call for action to improve the situation
- get input for my implementation project (which is at https://github.com/ugroempi/CAs)

Focus:

- solely on CAs and how to best make them available
- ignore many practical and very important challenges

CA notation in this talk

- N rows for the test runs
- k columns for the test variables
- v the number of levels for each variable in uniform CAs,
  - or $v_1, \dots, v_k$ for the k variables in mixed level CAs
- t strength, i.e.

  all possible tuples of level combinations **for any set of t columns**

  are covered at least once

Contents

1 Introduction

2 **Current status**

3 Implementation of mathematical constructions

4 Future of Colbourn tables and corresponding CAs

5 Call for input

6 References

Available catalogued CAs

Easy to find via https://csrc.nist.gov/projects/automated-combinatorial-testing-for-software:

- **NIST library** of uniform CAs ([4]):
  21 964 **actual arrays** ([4]; individually zipped CAs)
  - easily available for UIPs
  - sometimes substantially larger than known best CAs
- the **Colbourn tables** ([1]) for uniform CAs:
  - best-known run sizes for given *t*, *k*, *v*
  - based on known CA constructions
  - **no actual arrays**, only brief source tags
  - currently gone, temporarily available at https://github.com/ugroempi/CAs/blob/main/ColbournTables.md
- catalogue of uniform CAs by **Torres-Jiménez** ([16]),
  339 arrays (3 to 6 levels for strength 2 and 2 levels for strength 3)
  on Feb 6 2025
  (unavailable most of the time)

Actionability of Colbourn tables

Colbourn tables ([1]) are heavily cited as a respected source for

- the smallest known N for given k, t, v
- with brief info on the successful construction, e.g., “Cyclotomy (Colbourn)”

They lack actionability:

- UIPs cannot obtain a design from them
- even users with some expertise need substantial effort for obtaining a particular CA

Contents

1 Introduction

2 Current status

3 Implementation of mathematical constructions

4 Future of Colbourn tables and corresponding CAs

5 Call for input

6 References

Open source implementation

Goal: implement in the R package CAs

- constructions for current best CAs according to Colbourn tables ([1])
  - prioritize by importance and ease of implementation
  - sources for constructions in the tables
    (so far only partly) identifiable with the help of [2]
    (pointers very welcome!)

- many further constructions – also mixed-level – to be researched, e.g., [12], [18] *(pointers very welcome!)*

- constructions based on combining / modifying existing CAs

Detail: implement cyclotomy construction from [13]

213 CAs from pure cyclotomy construction ([13]) in Colbourn tables

[13] provides several related constructions (1, 2, 3, 3a, 3b, 4, 4a, 4b) that use “cyclotomic start vectors”

Table: Cyclotomy-based constructions in the Colbourn tables, source entry versus *v*

	2	3	4	6	7	8	10	11	12	14	18
Cyclotomy (Colbourn)	20	80	12	28	2	10	2	1	3	4	1
Cyclotomy (Colbourn) fuse	.	.	.	.	.	.	1	.	.	.	.
Cyclotomy (Colbourn) postop NCK	1	.	.	.	.	.	.	.	.	.	.
Cyclotomy (Torres-Jimenez)	37	3	.	2	1	.	1	.	.	.	.

Detail: implement cyclotomy construction (2)

Implementation uses

- **Galois field** GF(q) for a **prime** or prime power q
  available from R package lhs ([14], based on [15])
- ω: pre-calculated first primitive for GF(q)
  - the powers of a primitive span all non-zero elements of the GF
- x: $q \times 1$ **cyclotomic vector** for GF(q) and order v ($q \mod v = 1$):
  $x = \left(0, \log_\omega(1) \mod v, \ldots, \log_\omega(q-1) \mod v\right)^\top$
- A: $q \times q$ matrix indexed with $i, j = 0, \ldots, q-1$,
  obtained from x via $a_{ij} = x_{j-i}$ (difference in GF(q))
- final CA: use A (construction 1) or modifications thereof

Detail: implement cyclotomy construction (3)

	Construction							
	1	**2**	**3**	**3a**	**3b**	**4**	**4a**	**4b**
k	q	q	q	$q+1$	$q+1$	q	$q+1$	$q+1$
N	k	$k+v-1$	kv	$(k-1)v$	$v(k+v-2)$	$v(k+1)$	kv	$v(k+v-2)$

specific construction (i.e., q and construction) inferable from triple (N, k, v)

- ambiguity between 3b and 4b harmless (use 4b)
- likewise additional ambiguity between 4a and 4b for $v = 2$

Desirable for Colbourn tables:
provide **prime power** q and **exact construction**, in addition to “Cyclotomy”

Detail: implement cyclotomy construction (4)

Importance of trust:

- UIP cannot check coverage of large CAs
  - full checks often prohibitive, e.g., *Michael Wagner with internal version of CAmetrics on 30 kernels: 3.5 hours for brute-force coverage check of a CA(1051,4,1051,3)*
  - can at least check column samples
    → can prevent gross mistakes but not smaller non-coverage problems
- code check runs with examples from [13]:
  some mistakes found, most severe one:
  constructions 3 and 3b of Table 2 appear to be systematically wrong

Desirable for Colbourn tables:
provide information on how the CA was obtained / verified when and by whom → supports trust

Contents

1 Introduction

2 Current status

3 Implementation of mathematical constructions

4 **Future of Colbourn tables and corresponding CAs**

5 Call for input

6 References

Preserve and Improve

Preserving and **keeping up-to-date** requires action (currently gone, and not entirely up-to-date anymore)

Improvement is also desirable:

- actionability: how to obtain array of best-known size
  - Ideal for UIPs: provide **actual arrays**
  - Ideal for experts: provide construction details, (pseudo)code, ...
- in support of trust: provide **references** and/or **verification details**

Call for action

- call for a community-based team, with a team lead

Ideas for the team's activities (coarse sketch)

- migrate Colbourn tables to a versioned repository, e.g., GitHub (*properly, not like my quick-and-dirty pointer page with externally-hosted tables*)
  - easy input by community, e.g., by submitting issues or pull requests
  - permanent storage (enhance by linking to some permanent archiving tool like *UNESCO software heritage repository* ?)
  - deprecated entries to remain accessible
- define (and communicate) **information to be included** with table entries (e.g., date , the array, ...)
  - including formats
  - may require differentiation / flexibility
- **assemble** the defined information for current table entries – with community help
- define and communicate **updating procedures**

Introduction

Current status

Implementation of mathematical constructions

Future of Colbourn tables and corresponding CAs

Call for input

References

How to provide arrays

- each array in a **single file** – systematically-named, perhaps without *N* in the name (like in [4])?
- possibly **compressed** in a widely accessible way – e.g., zip (like in [4])
- with a **common file format** that could permit a substantial amount of **comments / background info** (like in the Torres-Jiménez catalogue, e.g., see the top rows of a CA with $N = 143$, $k = 199$, $v = 2$, $t = 4$.

```
C Criteria = 0
C Removed Row = 4
C Missings = 0
C Uncovering Columns = 0
C
143 199 2^199 4
1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 1 1
0 1 0 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0 0 0 1 0 1
0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 1 0 0
1 0 0 1 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0
```

Contents

1 Introduction

2 Current status

3 Implementation of mathematical constructions

4 Future of Colbourn tables and corresponding CAs

5 Call for input

6 References

Call for input

Introduction
Current status
Implementation of mathematical constructions
Future of Colbourn tables and corresponding CAs
Call for input
References

on my open source R implementation
of mathematical algorithms for CA constructions
https://github.com/ugroempi/CAs

- hints about promising mathematical constructions
- code that can be integrated / copied
- feedback, wishes
- use cases and test cases
- ...

Contents

1 Introduction

2 Current status

3 Implementation of mathematical constructions

4 Future of Colbourn tables and corresponding CAs

5 Call for input

6 **References**

References I

Introduction
Current status
Implementation of mathematical constructions
Future of Colbourn tables and corresponding CAs
Call for input
References

[1] C. J. Colbourn, “Covering array tables: 2 ≤v ≤25, 2 ≤t≤6, t≤k ≤10000, 2005–23.” [Online]. Available: https://www.public.asu .edu /~ccolbou/src /tabby

[2] J. Torres-Jimenez, I. Izquierdo-Marquez, and H. Avila-George, “Methods to construct uniform covering arrays,” *IEEE Access*, vol. 7, pp. 42774–42797, 2019, doi: 10.1109/ACCESS.2019.2907057.

[3] M. Leithner, A. Bombarda, M. Wagner, A. Gargantini, and D. E. Simos, “State of the CArt: evaluating covering array generators at scale,” *Int J Softw Tools Technol Transfer*, vol. 26, no. 3, pp. 301–326, Jun. 2024, doi: 10.1007/s10009-024-00745-2.

[4] “NIST Covering Array Tables.” Accessed: Nov. 10, 2024. [Online]. Available: https://math.nist.gov/coveringarrays/

[5] 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)*, Oct. 2020, pp. 191–200. doi: 10.1109/ICSTW50294.2020.00041.

[6] JMP Statistical Discovery LLC, “Design of experiments guide.” Section “Covering Arrays,” 2024. Available: https://www.jmp.com/support/help/en/18.0/index.shtml#page/jmp/covering-arrays.shtml

References II

[7] A. Dwyer and B. Stevens, *Covering Arrays (CA) - Combinatorics*. Accessed: Feb. 09, 2025. [Online]. Available: https://doc.sagemath.org/html/en/reference//combinat/sage/combinat/designs/covering_array.html#sage.combinat.designs.covering_array.covering_array

[8] A. Gargantini and M. Radavelli, "Migrating Combinatorial Interaction Test Modeling and Generation to the Web," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Vasteras: IEEE, Apr. 2018, pp. 308–317. doi: 10.1109/ICSTW.2018.00066.

[9] M. Leithner, K. Kleine, and D. E. Simos, "CAMETRICS: A Tool for Advanced Combinatorial Analysis and Measurement of Test Sets," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Apr. 2018, pp. 318–327. doi: 10.1109/ICSTW.2018.00067.

[10] D. J. Kleitman and J. Spencer, "Families of k-independent sets," *Discrete Mathematics*, vol. 6, no. 3, pp. 255–262, Jan. 1973, doi: 10.1016/0012-365X(73)90098-8.

[11] G. O. H. Katona, "Two applications (for search theory and truth functions) of Sperner type theorems," *Period Math Hung*, vol. 3, no. 1, pp. 19–26, Mar. 1973, doi: 10.1007/BF02018457.

[12] Y. Akhtar, C. J. Colbourn, and V. R. Syrotiuk, "Mixed-level covering, locating, and detecting arrays via cyclotomy," in *Combinatorics, graph theory and computing*, F. Hoffman, S. Holliday, Z. Rosen, F. Shahrokhi, and J. Wierman, Eds., Cham: Springer International Publishing, 2024, pp. 37–50. doi: 10.1007/978-3-031-52969-6_4.

References III

[13] C. J. Colbourn, “Covering arrays from cyclotomy,” *Des. Codes Cryptogr.*, vol. 55, no. 2, pp. 201–219, May 2010, doi: 10.1007/s10623-009-9333-8.

[14] R. Carnell, “lhs: Latin Hypercube Samples.” p. 1.2.0, 2024. doi: 10.32614/CRAN.package.lhs.

[15] A. Owen, *Orthogonal Arrays for Computer Experiments, Visualizations, and Integration in high dimensions: A C library*. (1994). Available: http://lib.stat.cmu.edu/designs/oa.c

[16] J. Torres-Jimenez, “Jose Torres-Jimenez Homepage,” Covering Arrays. Accessed: Feb. 06, 2025. [Online]. Available: https://www.tamps.cinvestav.mx/~oc/

[17] L. Yu, Y. Lei, R. N. Kacker, and D. R. Kuhn, “ACTS: A Combinatorial Test Generation Tool,” in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, Luxembourg, Luxembourg: IEEE, Mar. 2013, pp. 370–375. doi: 10.1109/ICST.2013.52.

[18] K. Shokri and L. Moura, “New Families of Strength-3 Covering Arrays Using Linear Feedback Shift Register Sequences,” *Journal of Combinatorial Designs*, vol. 33, no. 4, pp. 156–171, 2025, doi: 10.1002/jcd.21963.