Constructions of Covering Arrays

Charles J. Colbourn
Computer Science and Engineering
Arizona State University, Tempe, AZ

Challenge: Deleting a Symbol

0	0	0	2
1	1	1	2
2	2	2	2
0	1	2	1
1	2	0	1
2	0	1	1
0	2	1	0
1	0	2	0
2	1	0	0

It is well known that

CAN(2,k,v) ≤
CAN(2,k,v-1) – 1.

Challenge: Deleting a Symbol

0	0	0	$_{2}$0
1	1	1	$_{2}$0
2	2	2	$_{2}$0
0	1	2	1
1	2	0	1
2	0	1	1
0	2	1	$_{0}$2
1	0	2	$_{0}$2
2	1	0	$_{0}$2

Proof 1:

Make the first row constant by renaming symbols.

Then delete it.

Challenge: Deleting a Symbol

0	0	0	*
1	1	1	*
*0	*0	*0	*
0	1	*0	1
1	*0	0	1
*0	0	1	1
0	*0	1	0
1	0	*0	0
*0	1	0	0

Proof 2:

Change all of largest symbol in each column to * = “don’t care”

Then fill in * with entries from first row.

Then delete first row.

Challenge: Deleting a Symbol

0	0	0	$_{2}0$
1	1	1	$_{2}0$
2	2	2	$_{2}0$
0	1	2	1
1	2	0	1
2	0	1	1
0	2	1	$_{0}2$
1	0	2	$_{0}2$
2	1	0	$_{0}2$

First rename symbols and delete first row.

Challenge: Deleting a Symbol

1	1	1	*
2	2	2	*
*	1	2	1
1	2	*	1
2	*	1	1
*	2	1	2
1	*	2	2
2	1	*	2

Second replace all elements in the deleted row by *

Challenge: Deleting a Symbol

1	1	1	*
2	2	2	*
1	1	2	1
1	2	1	1
2	1	1	1
1	2	1	2
1	1	2	2
2	1	1	2

Now move top row elements into * positions and delete top row.

ARIZONA STATE UNIVERSITY

Challenge: Deleting a Symbol

2	2	2	*
1	1	2	1
1	2	1	1
2	1	1	1
1	2	1	2
1	1	2	2
2	1	1	2

This works in general and shows that

$$\mathrm{CAN}(2,k,v) \leq \mathrm{CAN}(2,k,v-1) - 2.$$

In fact it works for mixed covering arrays by removing one level from each factor.

Challenge: Deleting a Symbol

Is it always the case for $k,v \geq 2$ that

$$\mathrm{CAN}(2,k,v) \leq \quad \mathrm{CAN}(2,k,v-1) - \text{\Huge 3}?$$

For mixed CAs too?

True for OAs from the projective plane.

A Testing Problem

- The user is presented with n parameters ("factors"), each having some finite number of values ("levels").
- The j'th factor has s_j levels; continuous factors are modelled by a finite number of intervals.
- Initially, we assume that levels for factors can be selected independently.

Covering Arrays

- A covering array is an N x k array.
- Symbols in column j are chosen from an alphabet of size s_j
- Choosing any N x t subarray, we find every possible 1 x t row occurring at least once; t is the strength of the array.
- Evidently, the number N of rows must be at least the product of the t largest factor level sizes

Covering Arrays

- In general this is not sufficient. For constant t > 1 and factor level sizes, the number of rows grows at least as quickly as log n.
- Indeed, even for t=2, every two columns of the covering array must be distinct
- and this alone suffices to obtain a log n lower bound.

Arizona State University

Covering Arrays

$CA_\lambda(N;t,k,v)$

– An *N x k* array where each *N x t* sub-array contains all ordered *t-sets* at least λ times.

CA(6;2,5,2)

0	1	1	1	1
1	0	1	0	0
0	1	0	0	0
1	0	0	1	1
0	0	0	0	1
1	1	0	1	0

Covering Arrays

- The goal, given k, t, and the s_j's, is to minimize N. Or given N, t, and the s_j's, to maximize k.

table (5) - GSview

File Edit Options View Orientation Media Help

3	4	9^G	5	11^D	7	12^S	9	13^S	10	14^S
	20	15^T	24	17^T	30	18^T	36	19^T	43	20^T
	74	21^P	94	23^P	134	24^P	174	25^P	194	26^P
	394	27^P	474	29^P	594	30^P	714	31^P	854	32^P
	1402	33^P	1796	35^P	2364	36^P	3030	37^P	3766	38^P
	6836	39^P	8238	41^P	10000	42				

File: table (5) 343, 602pt Page: "1" 1 of 3

Covering Arrays

- Research on the problem has fallen into four main categories:
  - lower bounds
  - combinatorial/algebraic constructions
    - direct methods
    - recursive methods
  - probabilistic asymptotic constructions
  - computational constructions
    - exact methods
    - heuristic methods

Basic Combinatorial Methods

- Consider the problem of constructing a covering array of strength two, with g levels per factor, and k factors.
- We could hope to have as few as g^2 rows (tests), and if this were to happen then every 2-tuple of values would occur exactly once (a stronger condition than ‘at least once’).
- If we strengthen the condition to ‘exactly once’, the covering array is an orthogonal array of index one.

Orthogonal Arrays

$OA_{\lambda}(N;t,k,v)$ -An *N x k* array where each *N x t* sub-array contains all ordered *t-sets* *exactly* λ times.

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

OA(8;3,4,2)

ARIZONA STATE UNIVERSITY

Orthogonal Arrays

- For strength two, an orthogonal array of index one with g symbols and k columns exists
  - only when $k \leq g+1$,
  - if $k \leq g+1$ and g is a power of a prime.
- For primes, form rows of the array by including $(i,j,i+j,i+2j,\ldots,i+(g-1)j)$ for all choices of i and j, doing arithmetic modulo g as needed.
- For prime powers, the symbols used are those of the finite field.
- For non-prime-powers, lots of open questions!

ARIZONA STATE UNIVERSITY

Direct Methods

- OAs provide a direct construction of covering arrays.
- Another direct technique chooses a group on g symbols, and forms a ‘base’ or ‘starter’ array which covers every orbit of t-tuples under the action of the group.
- Then applying the action of the group to the starter array and retaining all distinct rows yields a covering array (typically exhibiting much symmetry as a consequence of the group action).

ASU Arizona State University

Direct Methods

- An example

$$(-,0,1,3,0,2,1,4)$$

- Form eight cyclic shifts
- Add a column of 0 entries
- Develop modulo 5
- Add the 6 constant rows (with – in last column) to get

$$CA(46;2,9,6)$$

Direct Methods

-	0	1	3	0	2	1	4	0
4	-	0	1	3	0	2	1	0
1	4	-	0	1	3	0	2	0
2	1	4	-	0	1	3	0	0
0	2	1	4	-	0	1	3	0
3	0	2	1	4	-	0	1	0
1	3	0	2	1	4	-	0	0
0	1	3	0	2	1	4	-	0

- Develop modulo 5
- Add 6 constant rows (with – in last column)

Direct Methods

- Stevens/Ling/Mendelsohn: From PG(2,q) delete a point to obtain a frame resolvable q-GDD of type $(q-1)^{(q+1)}$. Extend a frame pc and fill in “don’t care” positions to get a CA(2,q+2,q-1) with q^2-1 rows.
- (C, 2005) Can be extended to get a CA(2,q+1+x,q-x) for all nonnegative x. Relies only on having a row with no twice-covered pair.

Direct Methods

- Sherwood: Rather than use the field as a group of symmetries, use partial test suites build from the field and a compact means of determining when t such partial suites cover all possibilities.
- Sherwood, Martirosyan, C (2006): many new constructions for t=3,4,5
- Walker, C (preprint): and for t=5,6,7.

Recursive Methods

- A simple example (the Roux (1987) method).

A	A
B	$\overline{B}$

A is a strength 3 covering array, 2 levels per factor.

B is a strength 2 covering array, 2 levels per factor.

The bottom contains complementary arrays.

The result is a strength 3 covering array.

Arizona State University

Generalizing Roux

- Extensions by
  - Chateauneuf/Kreher (2001) to t=3, all g
  - Cohen/C/Ling (2004) to t=3, adjoining more than two copies, all g
  - Hartman/Raskin (2004) to t=4
  - Martirosyan/Tran Van Trung (2004) to all t under certain assumptions
  - Martirosyan/C (2005) to all t, all g.
  - C/Martirosyan/Trung/Walker(2006) for t=3, t=4.

Roux for two

- Prior to the Roux construction for $t \geq 3$, Poljak and Tuza had studied a direct product construction when $t=2$.
- This forms the basis of methods of Williams, Stevens, and Cohen & Fredman.

Roux for two

- Let A be a CA(N;2,k,v) and B a CA(M;2,f,v)

A	A	………	A
$b_1b_1b_1b_1$	$b_2b_2b_2b_2$	………	$b_fb_fb_fb_f$

is a CA(N+M;2,kf,v).

Roux for two

- Stevens showed that when each array has v constant rows, the resulting array has v duplicated rows and hence v rows can be removed.
- A recent extension (CMMSSY, 2006) shows that even when the arrays have “nearly constant” rows, again v rows can be eliminated.
- And an extension to mixed CAs.

Roux for two

- Let O be the all zero matrix
- Let C be a matrix with v rows, all of which are constant and distinct
- An SCA(N;2,k,v) A looks like

$$\left[\begin{array}{c|c} A1 & A2 \\ \hline C & O \end{array}\right]$$

Roux for two

Let A be a SCA(N;2,k,v), B a SCA(M;2,f,v) minus v rows forming C,O

A1 A2	A1 A2	………		A1
$b_1b_1b_1b_1$	$b_2b_2b_2b_2$	………		$b_fb_fb_fb_f$
C O	C O		O	O

has M+N-v rows

PHF and Turan Families

- Of particular note, but not enough time to discuss in detail:
  - Bierbrauer/Schellwat (1999): use a “perfect hash family” of strength t whose number of symbols equals the number of columns of the CA. Substitute columns for symbols. Asymptotically the best thing since sliced bread.
  - Hartman (2002): Turan families used much like above but more accurate for arrays with few symbols.

Four Values Per Factor

table (5) - GSview

File Edit Options View Orientation Media Help

4	5	16^G	6	19^D	7	21^S	8	22^S	10	24^S
	11	25^S	12	26^S	14	27^S	24	28^P	29	31^P
	30	32^P	34	33^P	38	34^P	40	35^P	49	36^P
	54	37^P	59	38^P	69	39^P	116	40^P	140	43^P
	144	44^P	164	45^P	184	46^P	192	47^P	236	48^P
	260	49^P	284	50^P	332	51^P	560	52^P	676	55^P
	696	56^P	792	57^P	888	58^P	928	59^P	1140	60^P
	1256	61^P	1372	62^P	1604	63^P	2704	64^P	3264	67^P
	3360	68^P	3824	69^P	4288	70^P	4480	71^P	5504	72^P
	6064	73^P	6624	74^P	7744	75^P	10000	76		

File: table (5) 413, 464pt Page: "1" 1 of 3

Six Values Per Factor

table (6) - GSview

File Edit Options View Orientation Media Help

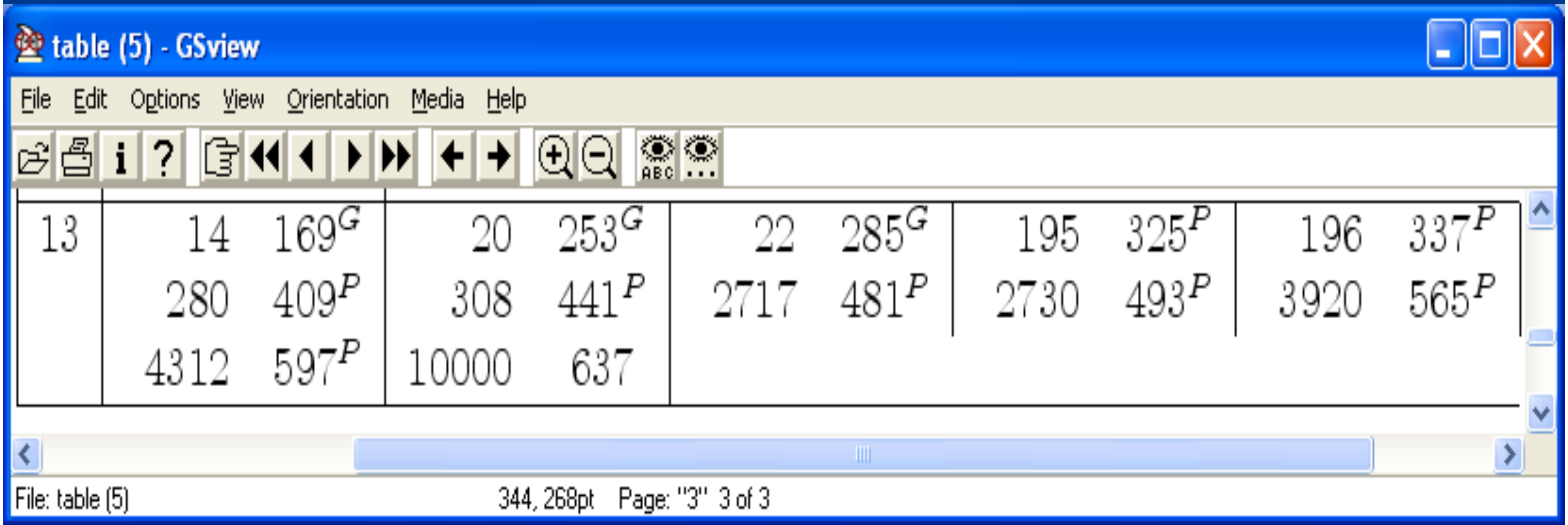
6	3	36^G	4	37^S	5	39^S	6	41^T	8	42^T
	9	46^D	10	51^D	11	55^S	12	56^S	13	58^S
	14	60^S	15	61^S	16	65^S	19	70^P	20	71^P
	24	72^P	26	73^P	32	74^P	34	75^P	40	76^P
	42	77^P	48	78^P	50	79^P	56	80^P	66	82^P
	72	84^P	80	86^P	81	88^P	89	91^P	90	92^P
	96	93^P	98	94^P	99	95^P	107	96^P	114	97^P
	120	98^P	125	100^P	134	101^P	135	102^P	139	105^P
	149	106^P	150	107^P	168	108^P	196	109^P	202	110^P
	260	111^P	288	114^P	304	115^P	336	116^P	360	117^P
	396	118^P	432	119^P	576	120^P	648	124^P	704	126^P
	729	128^P	784	131^P	810	133^P	864	134^P	880	135^P
	944	136^P	960	137^P	1024	138^P	1080	139^P	1104	140^P
	1184	141^P	1200	142^P	1215	143^P	1300	145^P	1364	146^P
	1560	147^P	1624	148^P	1820	149^P	2144	151^P	2304	152^P
	2448	153^P	2880	154^P	3024	155^P	3456	156^P	3600	157^P
	4032	158^P	4752	160^P	5184	162^P	5760	164^P	6144	166^P
	6480	168^P	6561	170^P	6912	171^P	7056	172^P	7209	173^P
	7704	174^P	8208	175^P	8640	176^P	9000	178^P	9648	179^P
	9720	180^P	10000	181						

File: table (6) 423, 424pt Page: "2" 2 of 3

Ten Values Per Factor

10	4	100^G	6	102^I	13	120^G	15	136^D	16	145^D
	17	154^D	18	163^D	19	172^D	20	174^C	21	190^D
	23	192^P	35	194^P	36	203^P	52	210^P	78	212^P
	169	230^P	195	246^P	208	255^P	221	264^P	225	271^P
	234	273^P	240	280^P	247	282^P	260	284^P	272	298^P
	273	300^P	312	302^P	468	304^P	676	320^P	1014	322^P
	1170	338^P	2197	340^P	2535	356^P	2704	365^P	2925	372^P
	3120	381^P	3328	390^P	3380	394^P	3536	399^P	3757	408^P
	3900	410^P	4056	412^P	6084	414^P	8788	430^P	10000	432

13 Values Per Factor



13	14 169^G	20 253^G	22 285^G	195 325^P	196 337^P
	280 409^P	308 441^P	2717 481^P	2730 493^P	3920 565^P
	4312 597^P	10000 637			

Tables

- For more tables than you can shake a stick at (and updates of the ones here), see
  - Colbourn (Disc Math, to appear) for t=2
  - C/M/T/W (DCC, to appear) for t=3, 4
  - Walker/C (preprint) for t=5
- We need better *general* direct constructions for small t, better recursions for large t.

ARIZONA STATE UNIVERSITY

Thanks