Sequence Covering Arrays

Charles J. Colbourn[1]

[1]School of Computing, Informatics, and Decision Systems Engineering
Arizona State University

Monash University, July 2012

Outline

Testing Event Sequences

- In some processes, for example in manufacturing, a set of tasks must be carried out in a certain sequence.
- But people are not good at following instructions, and sometimes do the steps (or events) in the wrong order.
- If certain subsets are done in the wrong order, this may cause the wrong behaviour.
- So we want to test the system to make sure that when users do some steps in the wrong order, *either* the process still works *or* the user gets an error message.

Sequence Covering Arrays

- Suppose that there are v steps in the process.
- Suppose that errors can be caused by performing some subset of t or fewer steps in a certain order.
- We want to ensure that for every subset of t or fewer steps, we perform the steps in each of the $t!$ orders at least once – note that in doing this, we still perform all v steps in some order.

Sequence Covering Arrays

- This problem was introduced by Kuhn, Higdon, Kacker, Lawrence, and Lei in April 2012 at the Workshop on Combinatorial Testing.
- They give a basic lower bound on the number of tests needed, and a heuristic algorithm for constructing tests.

Sequence Covering Arrays

- A t-subpermutation of $\{0, \ldots, v-1\}$ is a t-tuple $(x_1, \ldots, x_t)$ with $x_i \in \{0, \ldots, v-1\}$ for $1 \leq i \leq t$, and $x_i \neq x_j$ when $i \neq j$.
- A permutation π of $\{0, \ldots, v-1\}$ *covers* the t-subpermutation $(x_1, \ldots, x_t)$ if $\pi^{-1}(x_i) < \pi^{-1}(x_j)$ whenever $i < j$.
- For example, with $v = 5$ and $t = 3$, $(4, 0, 3)$ is a 3-subpermutation that is covered by the permutation 4 2 0 3 1.

Sequence Covering Arrays

- A *sequence covering array* of *order* v and *strength* t is a set $\Pi = \{\pi_1, \ldots, \pi_N\}$ where π_i is a permutation of $\{0, \ldots, v-1\}$, and every t-subpermutation of $\{0, \ldots, v-1\}$ is covered by at least one of the permutations $\{\pi_1, \ldots, \pi_N\}$.
- Call one a SeqCA($N; t, v$).

Sequence Covering Arrays
Example

SeqCA(9;3,7) – $t = 3$, $v = 7$, $N = 9$

0	5	6	4	3	2	1
2	1	6	5	0	3	4
3	4	5	1	2	0	6
6	1	2	4	3	0	5
0	1	4	3	6	2	5
5	2	3	4	6	0	1
3	6	1	5	0	2	4
4	0	1	2	5	6	3
6	2	5	1	3	4	0

Sequence Covering Arrays
The Existence Question

- Given t and v, what is the smallest N for which a SeqCA(N; t, v) exists?
- Call this number SeqCAN(t, v).
  - SeqCAN$(t, v) \geq t!$
  - SeqCAN$(t, v) \leq \binom{v}{t} t!$
- SeqCAN$(2, v) = 2$ for all $v \geq 2$ – Just take the identity permutation and its reversal!

Sequence Covering Arrays

Existence for $t > 2$

- SeqCAN$(3, v) \geq 6$, but it is not the case that SeqCAN$(3, v) = 6$ in general.
- To see why, we develop a connection with the usual notion of covering arrays, which we introduce next.

Covering Array. Definition

- Let N, k, t, and v be positive integers.
- Let C be an $N \times k$ array with entries from an alphabet Σ of size v; we typically take $\Sigma = \{0, \ldots, v-1\}$.
- When $(\nu_1, \ldots, \nu_t)$ is a t-tuple with $\nu_i \in \Sigma$ for $1 \leq i \leq t$, $(c_1, \ldots, c_t)$ is a tuple of t column indices ($c_i \in \{1, \ldots, k\}$), and $c_i \neq c_j$ whenever $\nu_i \neq \nu_j$, the t-tuple $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ is a *t-way interaction*.
- The array *covers* the t-way interaction $\{(c_i, \nu_i) : 1 \leq i \leq t\}$ if, in at least one row ρ of C, the entry in row ρ and column c_i is ν_i for $1 \leq i \leq t$.
- Array C is a *covering array* CA$(N; t, k, v)$ of *strength t* when every t-way interaction is covered.

Covering Array

CA(13;3,10,2)

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

Covering Array

- CAN(t, k, v) is the minimum N for which a CA$(N; t, k, v)$ exists.
- The basic goal is to minimize CAN(t, k, v).
- It is easy to establish that, when t and v are both fixed, CAN(t, k, v) is $O(\log k)$.

Lower Bounds for Sequence Covering Arrays

Theorem

Let k and t be integers satisfying $k \geq t \geq 3$. Whenever $0 \leq a < t$, the size of a sequence covering array for k events with strength t is at least

$$a!CAN(t-a, k-a, a+1)$$

Lower Bounds for Sequence Covering Arrays

- Choose any a events.
- For each ordering $(e_1, \dots, e_a)$ of the a events, select the permutations of the sequence covering array in which the a selected events appear in the chosen order; suppose that there are n such permutations.
- Form an $n \times (k - a)$ array A whose columns are indexed by the remaining events, and whose rows are indexed by the permutations selected.
- In the row indexed by π and the column indexed by event e,
  - place 0 if $\pi(e) < \pi(e_1)$
  - place a if $\pi(e) > \pi(e_a)$
  - otherwise, place i when $\pi(e_i) < \pi(e) < \pi(e_{i+1})$.

Lower Bound

Example

SeqCA(9;3,7) – $t = 3$, $v = 7$, $N = 9$
take $a = 1$ and use symbol 6

0	5	6	4	3	2	1	0	1	1	1	1	0
2	1	6	5	0	3	4	1	0	0	1	1	1
3	4	5	1	2	0	6	0	0	0	0	0	0
6	1	2	4	3	0	5	1	1	1	1	1	1
0	1	4	3	6	2	5	0	0	1	0	0	1
5	2	3	4	6	0	1	1	1	0	0	0	0
3	6	1	5	0	2	4	1	1	1	0	1	1
4	0	1	2	5	6	3	0	0	0	1	0	0
6	2	5	1	3	4	0	1	1	1	1	1	1

Lower Bounds for Sequence Covering Arrays

- The theorem can give better estimates.
- For $t = 4$, we obtain the lower bounds
  - $\text{SeqCAN}(4, v) \geq \text{CAN}(4, v, 1)$ by taking $a = 0$,
  - $\text{SeqCAN}(4, v) \geq \text{CAN}(3, v-1, 2)$ by taking $a = 1$,
  - $\text{SeqCAN}(4, v) \geq 2\text{CAN}(2, v-2, 3)$ by taking $a = 2$,
  - $\text{SeqCAN}(4, v) \geq 6\text{CAN}(1, v-3, 4) = 24$ by taking $a = 3$.
- Taking $a = 0$ always gives a trivial bound.

Upper Bounds for Sequence Covering Arrays

Random Arrays

- Suppose that you pick a permutation of $\{0, \ldots, v-1\}$ uniformly at random.
- Any specific t-subpermutation is covered with probability $\frac{1}{t!}$.
- And it fails to be covered with probability $\frac{t!-1}{t!}$.
- So if you pick N permutations of $\{0, \ldots, v-1\}$ uniformly at random and independently, any specific t-subpermutation is covered with probability $1 - \left(\frac{t!-1}{t!}\right)^N$.

Upper Bounds for Sequence Covering Arrays

Random Arrays

- There are $\frac{v!}{(v-t)!}$ t-subpermutations.
- When N permutations are chosen, each subpermutation is *not* covered with probability $\left(\frac{t!-1}{t!}\right)^N$.
- So the probability that at least one subpermutation is not covered is at most $\frac{v!}{(v-t)!}\left(\frac{t!-1}{t!}\right)^N$.

Upper Bounds for Sequence Covering Arrays

Random Arrays

- When $\frac{v!}{(v-t)!}\left(\frac{t!-1}{t!}\right)^N < 1$, a SeqCA$(N; t, v)$ must exist!
- So when t is fixed, this gives an O$(\log v)$ upper bound on the number of permutations needed.

Upper Bounds for Sequence Covering Arrays

Greedy Random Arrays

- Instead consider generating the set of permutations *one permutation at a time*.
- Idea: Always pick the next permutation so that it covers the largest possible number of as-yet-uncovered t-subpermutations.
- Suppose that after i permutations have already been chosen, there remain R_i t-subpermutations to be covered.
- Then $R_0 = \frac{v!}{(v-t)!}$.

Upper Bounds for Sequence Covering Arrays

Greedy Random Arrays

- Consider selecting the $(i+1)$st permutation.
- If we choose it uniformly at random, then as before every (as-yet-uncovered) t-subpermutation is covered by the chosen permutation with probability $\frac{1}{t!}$.
- So using the linearity of expectations ("the sum of the expectations is the expectation of the sum"), the expected number of t-subpermutations covered *for the first time* by the chosen permutation is $\frac{R_i}{t!}$.

Upper Bounds for Sequence Covering Arrays

Greedy Random Arrays

- But then some permutation covers at least $\frac{R_i}{t!}$ t-subpermutations for the first time, and hence

$$R_{i+1} \leq R_i - \frac{R_i}{t!}.$$

- Combine with $R_0 = \frac{v!}{(v-t)!}$ to get

$$R_i \leq \left(\frac{t!-1}{t!}\right)^i \frac{v!}{(v-t)!}.$$

- When $R_N < 1$, a SeqCA$(N; t, v)$ exists – and the greedy method guarantees to find one!

Stein-Lovász-Johnson

- The greedy strategy has proved remarkably successful for a variety of combinatorial covering problems.
- Most surprising is that in many cases it has been shown that we can select the next row efficiently (in time polynomial in the number of columns when the strength t is fixed).
- Such efficient methods have been found for
  - covering arrays (Bryce and Colbourn (2007, 2009))
  - perfect hash families (Colbourn (2008))
  - separating, distributing, strengthening, scattering, ... hash families (Colbourn (2011); Colbourn, Horsley, and Syrotiuk (2012))

Stein-Lovász-Johnson

- The key observations in these efficient methods are
  - It is enough to choose the next "row" so that it covers *at least the average number of as-yet-uncovered "entities"*
  - We can fill in the row one entry at a time so that the expected number covered by any completion of the row never decreases, and what we need to do is to find an entry – efficiently – that does not decrease this expected number.

- We implemented this method, and report some computational results here.

Upper Bounds for Sequence Covering Arrays

Existence Results

Events	$t = 3$ U	U_R	O	D_R	D	Other
4	12	12	-	8	6	
5	17	16	8	10	8	7
6	20	18	10	10	8	
7	23	22	12	12	9	8
8	26	24	12	12	10	9
9	28	26	14	12	11	10
10	30	28	14	14	12	10
11	32	30	14	14	12	
12	33	30	16	14	13	
13	35	32	16	16	13	
14	36	34	16	16	14	
15	37	34	18	16	14	
16	39	36	18	16	15	
17	40	36	20	18	15	
18	41	38	20	18	16	
19	42	38	22	18	16	
20	42	38	22	18	16	

Events	$t = 3$ U	U_R	O	D_R	D
21	43	40	22	18	17
22	44	40	22	20	17
23	45	40	24	20	17
24	46	42	24	20	17
25	46	42	24	20	18
26	47	42	24	20	18
27	48	44	26	20	18
28	48	44	26	20	18
29	49	44	26	22	19
30	49	46	26	22	19
40	54	50	32	24	21
50	58	52	34	26	23
60	61	56	38	26	24
70	64	58	40	28	25
80	66	60	42	30	26
90	68	62	-	30	27

Events	$t=4$					$t=5$			
	U	U_R	O	D_R	D	U	U_R	D_R	D
4	24	24	24	24	24				
5	54	54	29	24	26	120	120	120	120
6	79	78	38	32	34	294	294	148	149
7	98	96	50	40	41	437	436	198	200
8	114	112	56	44	47	552	550	242	243
9	128	126	68	50	52	648	646	282	284
10	140	138	72	56	57	731	728	318	322
11	151	148	78	60	61	803	800	354	356
12	160	158	86	64	66	868	864	384	386
13	169	166	92	70	71	926	922	416	419
14	177	174	100	74	73	978	976	446	448
15	184	180	108	78	78	1027	1024	470	475
16	191	188	112	80	81	1072	1068	496	501
17	197	194	118	84	84	1113	1110	518	521
18	203	200	122	86	86	1152	1148	540	547
19	209	204	128	90	91	1189	1184	560	570
20	214	210	134	92	92	1223	1218	582	590
21	219	214	134	96	95	1256	1252	600	610
22	224	220	140	98	97	1286	1282	622	629
23	228	224	146	98	99	1316	1310	636	646
24	232	228	146	102	101	1344	1338	654	665
25	236	232	152	104	104	1370	1366	674	682
26	240	236	158	106	105	1396	1390	688	698
27	244	240	160	108	107	1420	1416	706	715
28	248	242	162	110	110	1444	1438	718	732
29	251	246	166	112	111	1466	1460	734	746
30	255	250	166	114	113	1488	1482	748	760
40	283	278	198	132	128	1671	1644		
50	305	298	214	146	141	1811	1804		
60	322	316	238	154	151	1924	1916		
70	337	330	250	166	160	2019	2012		
80	350	342	264	174	168	2101	2092		
90	361	354	-	180	176	2173	2164		

Open Problems # 1

- Can we explain why including the reversal of a permutation seems like a "good idea" but does not necessarily lead to the best result?
- Can we determine when SeqCAN$(t, v) = t!$? This is related to *Directed Steiner t-designs*.

Open Problems # 2

- Find combinatorial direct constructions for sequence covering arrays.
- Find combinatorial recursive constructions for sequence covering arrays. (There is some recent progress on this!)

Open Problems # 3

- Can SeqCAN$(3, v)$ be determined exactly?
- Can we construct a sequence covering array from a covering array? What conditions are necessary? sufficient?