On the Structure of Small Strength-2 Covering Arrays

Janne I. Kokkala*
Department of Communications and Networking
Aalto University School of Electrical Engineering
P.O. Box 15400, 00076 Aalto, Finland

Karen Meagher†
Department of Mathematics and Statistics
University of Regina
Regina, SK, S4S 0A2
Canada

Reza Naserasr‡
Institut de Recherche en Informatique Fondamentale
Bâtiment Sophie Germain
8 place Aurélie Nemours
75013 Paris, France

Kari J. Nurmela§, Patric R. J. Östergård¶
Department of Communications and Networking
Aalto University School of Electrical Engineering
P.O. Box 15400, 00076 Aalto, Finland

Brett Stevens‖
School of Mathematics and Statistics
Carleton University
1125 Colonel By Drive
Ottawa, ON, K1S 5B6
Canada

Abstract

A covering array $\mathrm{CA}(N; t, k, v)$ of strength t is an $N \times k$ array of symbols from an alphabet of size v such that in every $N \times t$ subarray, every t-tuple occurs in at least one row. A covering array is *optimal* if it has the smallest possible N for given t, k, and v, and *uniform* if every symbol occurs $\lfloor N/v \rfloor$ or $\lceil N/v \rceil$ times in every column. Prior to this paper the only known optimal covering arrays for $t = 2$ were orthogonal arrays, covering arrays with $v = 2$ constructed from Sperner's Theorem and the Erdős-Ko-Rado Theorem, and eleven other parameter sets with $v > 2$ and $N > v^2$. In all these cases, there is a uniform covering array with the optimal size. It has been conjectured that there exists a uniform covering array of optimal size for all parameters. In this paper a new lower bound as well as structural constraints for small uniform strength-2 covering arrays are given. Moreover, covering arrays with small parameters are studied computationally. The size of an optimal strength-2 covering array with $v > 2$ and $N > v^2$ is now known for 21 parameter sets. Our constructive results continue to support the conjecture.

1 Introduction

A covering array $\mathrm{CA}(N; t, k, v)$ of strength t is an $N \times k$ array of symbols from an alphabet of size v such that in every $N \times t$ subarray, every t-tuple occurs in at least one row. We will use $Z_v = \{0, 1, \ldots, v-1\}$ as the alphabet for all of our covering arrays. A covering array is *optimal* if it has the smallest possible N for given t, k, and v, and *uniform* if every symbol occurs either $\lfloor N/v \rfloor$ or $\lceil N/v \rceil$ times in every column. A uniform $\mathrm{CA}(N; t, k, v)$ is denoted by $\mathrm{UCA}(N; t, k, v)$. The smallest value of N for which a $\mathrm{CA}(N; t, k, v)$ (respectively $\mathrm{UCA}(N; t, k, v)$) exists is denoted by $\mathrm{CAN}(t, k, v)$ (respectively $\mathrm{UCAN}(t, k, v)$).

Covering arrays are extensively studied designs with many applications. There are several surveys of covering arrays [4, 11, 19]; for more recent studies see [1, 5, 6, 10, 34, 43, 45]. Uniform covering arrays are particularly useful since

*Supported by the Aalto ELEC Doctoral School, Nokia Foundation, and Academy of Finland, Project #289002. Present address: Department of Theoretical Computer Science, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden.

†Supported in part by an NSERC discovery grant.

‡ANR-17-CE40-0022

§Present address: Mankkaanmalmi 8 A, 02180 Espoo, Finland

¶Supported in part by the Academy of Finland, Project #289002.

‖Supported in part by an NSERC discovery grant.

they are used in some constructions to create larger covering arrays [8, 10, 33, 46]. In this work, we only consider strength-2 covering arrays; thus we omit the parameter t for brevity, and write $\mathrm{CA}(N; k, v)$ and $\mathrm{UCA}(N; k, v)$ instead of $\mathrm{CA}(N; 2, k, v)$ and $\mathrm{UCA}(N; 2, k, v)$, respectively. We also use $\mathrm{CAN}(k, v)$ and $\mathrm{UCAN}(k, v)$ for $\mathrm{CAN}(2, k, v)$ and $\mathrm{UCAN}(2, k, v)$.

A covering array with $N = v^2$ is an orthogonal array and it is necessarily both uniform and optimal. For $v = 2$ it is known ([16, 17]) that $\mathrm{CAN}(k, 2) = n$, where

$$\binom{n-2}{\lceil (n-1)/2 \rceil} < k \leq \binom{n-1}{\lceil n/2 \rceil}.$$

Moreover, for $v = 2$ and all k, there is a uniform covering array of optimal size, so $\mathrm{UCAN}(k, 2) = \mathrm{CAN}(k, 2)$ (this is a consequence of a graph homomorphism and graph core result [23, Theorem 5]).

Prior to this work, as far as we have been able to verify, the other optimal values known when $N > v^2$ were $\mathrm{CAN}(5, 3) = 11$, $\mathrm{CAN}(6, 3) = \mathrm{CAN}(7, 3) = 12$, $\mathrm{CAN}(8, 3) = \mathrm{CAN}(9, 3) = 13$, $\mathrm{CAN}(10, 3) = 14$, $\mathrm{CAN}(6, 4) = 19$, $\mathrm{CAN}(7, 4) = 21$, $\mathrm{CAN}(7, 5) = 29$, $\mathrm{CAN}(4, 6) = 37$, and $\mathrm{CAN}(5, 6) = 39$ (see Table 2 in the current paper for references). In all these cases, there exists an optimal covering array that is also uniform. In fact, to date there has not been a single set of parameters found for which none of the optimal covering arrays is uniform. This has led the second and sixth author of the current paper to make the following conjecture [23, Conjecture 1].

Conjecture 1. *If there exists a* $\mathrm{CA}(N; k, v)$ *then there exists a* $\mathrm{UCA}(N; k, v)$.

Recently, Torres-Jimenez [42] found examples of optimal, but not uniform, covering arrays with the additional property that the array has the maximum number of columns (maximum k) for the given number of rows (given N). One generalization of covering arrays is covering arrays avoiding forbidden edges where certain pairs of symbols in certain columns are forbidden [9]. There exists an arc-transitive 4-partite graph where the unique optimal covering array avoiding the edges of the graph cannot be uniform [36]. This does not refute the conjecture but it does show that placing even highly symmetric constraints on covering arrays can force non-uniformity of optimal arrays.

An analogous problem has also been studied for covering and packing (error-correcting) codes. For binary covering codes, there are sets of parameters for which all optimal codes are nonuniform [30]. For binary error-correcting codes, there are even sets of parameters for which all optimal codes have a nonuniform distribution of coordinate values in all coordinates [31].

The main challenge in studying Conjecture 1—in searching for a counter-example—is to determine $\mathrm{CAN}(k, v)$. This can be done via a lower bounds

and a constructive upper bound that meet. There are some well-known constructions for covering arrays. Specific covering arrays can be found by metaheuristic search techniques [3, 27], using constraint programming models [12] or by applying post-optimization techniques to known constructions [25]. In practice, however, strong enough bounds are in general available only for limited sets of parameters [12, 37]

A new lower bound on the size of covering arrays is proved in this paper. Analytical methods can be augmented with computational techniques, which will be utilized in the current work to determine $\mathrm{CAN}(k, v)$ up to the limits set by the available algorithms and computational resources.

In this paper, new lower bounds and structural constraints on uniform covering arrays are given in Section 2. Computational methods, including exhaustive search and classification procedures, are described in Section 3. A central aspect of efficient exhaustive search is avoiding finding different copies of the same array. Equivalent covering arrays, made from permuting columns, rows or symbols, are called *isomorphs* and the process of not spending computational time to find an array equivalent to one already found is called *isomorph rejection*. These concepts are discussed in more detail in Section 3. An extensive table of classification results is given in Section 3.2. Finally, our computational results are discussed in Section 4, which also contains updated tables of bounds on $\mathrm{CAN}(k, v)$ and $\mathrm{UCAN}(k, v)$ for $4 \leq k \leq 10$ and $3 \leq v \leq 6$.

2 Bounds for small covering arrays

2.1 A lower bound for uniform covering arrays

Theorem 1. *Let C be a $\mathrm{UCA}(N; k, v)$. Let $d = \lfloor N/v \rfloor$ and $i = N - vd$. Then*

$$(k^2-3k+2v)N^2-v(k(2v-1)-2)(k-1)N+k(k(v^4-v^3+vi-i^2)-(v^4-v^3+3vi-3i^2)) \geq 0$$

and a necessary condition for equality is that every pair of rows in C agree in at least one and at most two columns.

Proof. Let C be a $\mathrm{UCA}(N; k, v)$, $d = \lfloor N/v \rfloor$, and $i = N - vd$. To arrive at the inequality, we will find an upper and a lower bound for the number of pairs of rows which agree in at least one position. An upper bound is the total number of pairs of rows, $\binom{N}{2}$.

To get a lower bound on the number of pairs of rows we introduce two new parameters. Define M_1 to be the number of triples (r, r', c) for which rows r

and r' agree in column c. Further define M_2 to be the number of quadruples (r, r', c, c') for which rows r and r' agree in columns c and c'. Then $M_1 - M_2$ is a lower bound on the number of pairs of rows which agree in at least one position. (Indeed, $M_1 - M_2$ consists of the first two terms in the summation using the principle of inclusion and exclusion.)

This gives us the bound

$$M_1 - M_2 \leq \binom{N}{2}, \tag{1}$$

which is tight if and only if every pair of rows in C agree in at least one and at most two columns.

Since the array is uniform, in every column there are i symbols which appear $d+1$ times and $v-i$ symbols which appear only d times. Thus the contribution to M_1 from any column is $i\binom{d+1}{2} + (v-i)\binom{d}{2}$ and the sum of these over all columns is

$$M_1 = k\left(i\binom{d+1}{2} + (v-i)\binom{d}{2}\right) = \frac{kd(N-v+i)}{2}. \tag{2}$$

Next, we find an upper bound for M_2. Consider columns c and c'. Let $\lambda_{x,y}^{c,c'}$ be the number of rows r such that $C_{r,c} = x$ and $C_{r,c'} = y$, and let $\mu_x^{c,c'}$ be the number of pairs of rows r and r' such that $C_{r,c} = C_{r',c} = x$ and $C_{r,c'} = C_{r',c'}$. It follows from the definition that

$$\mu_x^{c,c'} = \sum_y \binom{\lambda_{x,y}^{c,c'}}{2}. \tag{3}$$

The number of pairs of rows that agree in columns c and c' is then $\sum_x \mu_x^{c,c'}$.

For each x, let m_x^c be the number of times x occurs in column c. Since $\sum_y \lambda_{x,y}^{c,c'} = m_x^c$ and $\lambda_{x,y}^{c,c'} \geq 1$ for all y, it can be seen that Equation (3) is maximized for each x when there is a y_x such that $\lambda_{x,y_x}^{c,c'} = m_x^c + 1 - v$, and $\lambda_{x,y}^{c,c'} = 1$ for all $y \neq y_x$. This gives

$$\sum_x \mu_x^{c,c'} \leq \sum_x \binom{m_x^c + 1 - v}{2} = i\binom{d+2-v}{2} + (v-i)\binom{d+1-v}{2}. \tag{4}$$

The last equality follows from the fact that i symbols occur $d+1$ times in column c, and $v-i$ symbols occur d times in column c. This bound is attained if and only if there is a permutation π of $\{0, 1, \ldots, v-1\}$ such that the number of times x appears in column c equals the number of times $\pi(x)$ appears in column c' for each x, and $\lambda_{x,y} = 1$ whenever $y \neq \pi(x)$.

Summing (4) over all pairs of columns gives us an upper bound for M_2,

$$M_2 = \sum_{c,c'} \sum_{x} \mu_x^{c,c'} \leq \frac{k(k-1)(d+1-v)(N-v^2+i)}{4}. \tag{5}$$

Applying (2) and (5) to (1) and multiplying both sides by $4v$ yields the bound from the theorem. □

This theorem is useful for small k.

Corollary 2. *If there exists a* UCA$(N; v+2, v)$*, then* $N \geq v^2+v-1$*. Further, if* $N = v^2 + v - 1$*, then every pair of rows must agree in either one or two positions, and in each pair of columns there are exactly* $v - 1$ *disjoint pairs of symbols that appear twice.*

Similarly we can apply Theorem 1 to covering arrays with few columns.

Corollary 3. *Assume that there exists a* UCA$(N; v + j, v)$*.*

1. *If* $j = 3$*, then* $N > v^2 + 3v/2 - 5/2$*. If additionally* $2 < v \leq 11$*, then* $N > v^2 + 3v/2 - 2$*.*
2. *If* $j = 4$ *then,* $N > v^2 + 2v - 5$*. If additionally* $v \leq 6$*, then* $N \geq v^2 + 2v - 4$*.*
3. *If* $j = 5$*, then* $N > v^2 + 7v/3 - 13/2$*.*
4. *If* $j = 6$*, then* $N > v^2 + 8v/3 - 21/2$*.*
5. *If* $j = 7$*, then* $N > v^2 + 3v - 15$*.*

In the previous corollary, there are similar improvements possible in the constant term in the lower bound on N when v is sufficiently small for $j \geq 5$. We only state the improved bounds for $j = 3$ and 4.

The form of the bound in Theorem 1 does not let us easily identify its behaviour as a function of k, but, by losing the accuracy given by the residue of N mod v, we can obtain a weaker bound that has a more directly computable form.

Corollary 4. *Let*

$$b = (2v-3)k^2 + (-2v+5)k + (-4v+2),$$
$$a = 2k^2 - 6k + 4v, \text{ and}$$
$$D = k^4 + (8v^2 - 16v + 2)k^3 + (-8v^3 + 24v - 3)k^2 + (8v^3 - 8v^2 - 8v - 4)k + 4.$$

If there exists a $\mathrm{UCA}(N; k, v)$*, then*

$$N \geq v\left(\frac{b+\sqrt{D}}{a}\right) \tag{6}$$

Proof. Consider a uniform covering array $\mathrm{CA}(N; k, v)$. Let N' be the least multiple of v that is at least N. This means that $N' \leq N + v - 1$ and the uniform $\mathrm{CA}(N; k, v)$ can be extended to a $\mathrm{CA}(N'; k, v)$ in which each column has each symbol occurring exactly N'/k times. Theorem 1 can be applied to the $\mathrm{CA}(N'; k, v)$, to get that

$$2vN'(N'-1) \geq k\left(2(N')(N'-v) - (k-1)(N'-v^2+v)(N'-v^2)\right).$$

This reduces to

$$0 \leq (k^2-3k+2v)(N')^2 - v(k-1)(2kv-k-2)N' - k(k-1)(v^4-v^3) \leq 0$$

Then N' must be bounded below by the quadratic's larger root. The result follows since $N' - (v-1) \leq N$. □

By taking the derivative of Inequality (6) with respect to k and approximating its roots we compute that this bound reaches it maximum at a value of k less than, but close to

$$k_{\max} \approx \frac{16v^2 - 20v - 15}{8v - 16} = 2v + \frac{3}{2} + \frac{9}{8v-16}.$$

The error in this approximation is less then 0.5 after $v = 16$. The value of the bound at this maximum point is approximately

$$N_v = \mathrm{UCAN}(2, k_{\max}, v) \geq 2.4142v - 1.17678 - \frac{3.5026v + 10.2260}{8v^2 + 8v - 4}.$$

That is, for $k > k_{\max}$ the value of the bound from Corollary 4 is smaller than N_v. Since we know that $\mathrm{UCAN}(2, k+1, v) \geq \mathrm{UCAN}(2, k, v)$, the bound from Corollary 4 loses its utility for any $k > k_{\max}$. The maximum useful k for the bound of Theorem 1 must also be close to this $k_{\max}$. In our classification results six uniform covering arrays meet the bound from Theorem 1. Five have $k = v + 2$ and one, $\mathrm{UCA}(21; 7, 4)$, has $k = v + 3$.

2.2 Constraints on covering arrays with $v + 2$ columns

The strongest structural conditions implied by equality in Theorem 1 happen when $k = v + 2$ and $N = v^2 + v - 1$. We further investigate covering arrays

with these parameters. First we introduce some notation. In a uniform covering array UCA$(v^2+v-1;k,v)$, in each column every symbol occurs either v times or $v+1$ times. An entry in a UCA$(N;k,v)$ is called a *high frequency entry* if the symbol in the entry occurs at least $v+1$ times in the entry's column.

Theorem 5. *Let C be a* UCA$(v^2+v-1;v+2,v)$ *and let a_i be the number of rows that contain exactly i high frequency entries. Then*

$$\sum_{i=0}^{v+2} a_i = v^2+v-1, \tag{7}$$

$$\sum_{i=0}^{v+2} i a_i = (v+2)(v-1)(v+1), \tag{8}$$

$$\sum_{i=0}^{v+2} i^2 a_i = (v+2)(v+1)^2(v-1). \tag{9}$$

Further, $a_0 \leq 1$, and $a_1 = a_2 = 0$.

Proof. Let C be a UCA$(v^2+v-1;v+2,v)$. For a column, c, denote by S_c the set of symbols in high frequency entries. We know $|S_c| = v-1$.

Equation (7) is established by simply counting the rows of C. Equation (8) is established by computing the cardinality of the set

$$\{(x,c,r) \mid 0 \leq c < v+2,\ C_{r,c} = x \in S_c\}.$$

Equation (9) is established by computing the cardinality of the set

$$\{(x,c,y,c',r) \mid 0 \leq c \neq c' < v+2,\ C_{r,c} = x \in S_c, C_{r,c'} = y \in S_{c'}\}.$$

There is exactly one symbol per column that is repeated exactly v times. So if two rows had no high frequency entries, then both rows would only contain the symbols that occur exactly v times. This would mean that a pair of such symbols is repeated and C could not be a covering array. Thus $a_0 \leq 1$.

To establish that $a_1 = a_2 = 0$, let r be a fixed row containing i symbols which appear $v+1$ times in their column. For any of the other v^2+v-2 rows b, let $\mu_{r,b}$ be the number of columns where rows r and b agree. Counting the flags (c,b) with $0 \leq c < v+2$, $b \neq r$ and $C_{b,c} = C_{r,c}$ we have

$$\sum_{b \neq r} \mu_{r,b} = v^2+v-2+i, \tag{10}$$

$$\overline{\mu} = \frac{v^2+v-2+i}{v^2+v-2}. \tag{11}$$

Counting the flags (c, c', b) such that $0 \leq c \neq c' < v+2$, $C_{b,c} = C_{r,c}$, and $C_{b,c'} = C_{r,c'}$ we have

$$\sum_{b \neq r} \binom{\mu_{r,b}}{2} \leq \binom{i}{2}, \tag{12}$$

(this follows since only high frequency entries can occur twice). Using Equation (10), this implies that

$$\sum_{b \neq r} \mu_{r,b}^2 \leq v^2 + v - 2 + i^2. \tag{13}$$

Now we get

$$\begin{aligned} 0 &\leq \sum_{b \neq r} (\mu_{r,b} - \overline{\mu})^2 \\ &= -\overline{\mu}^2(v^2 + v - 2) + \sum_{b \neq r} \mu_{r,b}^2 \\ &\leq i\frac{i(v^2 + v - 3) - (2v^2 + 2v - 4)}{v^2 + v - 2}. \end{aligned}$$

Which implies that $i = 0$ or $i \geq \lceil 2 + 2/(v^2 + v - 3) \rceil = 3$. □

Corollary 6. *Let C be a* $\mathrm{UCA}(v^2 + v - 1; v + 2, v)$ *and let a_i be the number of rows that have exactly i high frequency entries. If $a_0 = 1$, then $a_{v+1} = v^2 + v - 2$ and $a_i = 0$ for all other i.*

Proof. If $a_0 = 1$ then (7), (8) and (9) imply that the average i is $\bar{i} = v + 1$. The variance in the distribution of $i \neq 0$ is equal to 0, since

$$\begin{aligned} \sum_{i=1}^{v+2} (i - \bar{i})^2 &= -\bar{i}^2(v+2)(v-1) + \sum_{i=1}^{v+2} i^2 a_i \\ &= (v+2)(v-1)(v+1)^2 - (v+2)(v-1)(v+1)^2. \end{aligned}$$

□

All this leaves open the possible existence of a $\mathrm{CA}(N; v+2, v)$ with $N < v^2 + v - 1$ (and thus smaller than those described in Corollary 3) if only the covering array is *not* uniform. However some constraints exist even in this case for $\mathrm{CA}(N; v+2, v)$. In [37], the following result is proved using a similar counting method.

Theorem 7. *Assume that there exists a* $\mathrm{CA}(N; k, v)$ *that has a row that contains at most two high frequency entries. If $k = v+2$, then $N \geq v^2+v-1$; and if $k \geq v + 3$, then $N \geq v^2 + v$.*

Assume that there exists a $\mathrm{CA}(N; k, v)$ *that has a row that contains at most three high frequency entries. If $k \geq v + 2$, then $N \geq v^2 + v - 1$.*

3 Classifying covering arrays

In all of our computer-aided studies of covering arrays, we fix the parameters of the array: the order, v; the degree, k; and the size, N. We further consider covering arrays as multisets of their rows. (Given two multisets, S and T, the *multiset sum* $S \uplus T$ is the set for which the multiplicity of each element is the sum of its multiplicities in S and T.) Two covering arrays are then said to be *equivalent* if one can be obtained from the other by a permutation of the columns and by column-wise permutations of the elements of Z_v. A transformation that maps a covering array C onto itself is an *automorphism*, and the set of all automorphisms form the *(full) automorphism group* of C, denoted by $\mathrm{Aut}(C)$.

Our computer search builds all inequivalent covering arrays with a given parameter set. This is done by isomorph rejection. We represent covering arrays as colored graphs (this is described below) and, using *nauty* [22], we determine the automorphism group for each of these graphs.

The colored graph G corresponding to a covering array C is constructed in the standard way [32]. The vertices of G will be colored with 2 colors to prevent a mapping between vertices of different colors. Colourings of graphs are the mechanism that *nauty* [22] uses to forbid mappings between vertices; they need not be proper colourings. First, G contains k disjoint copies of the complete graph of order v; all of these vertices are colored with the first color. These vertices represent the entries in the columns of the CA; the ith copy of the complete graph corresponds to the ith column and the jth vertex in each copy corresponds to the symbol j in that column. Further, G contains N vertices, representing the rows of C, all colored with the second color. Each such vertex is connected to the k vertices corresponding to the column-symbol pairs that occur in that row. Note that the obvious homomorphism $\mathrm{Aut}(G) \to \mathrm{Aut}(C)$ has a nontrivial kernel if there are duplicate rows.

Our main method, presented in Section 3.1, constructs one representative from each equivalence class of covering arrays, $\mathrm{CA}(N; k, v)$. This is done by starting with a set of representatives of the equivalence classes of covering arrays $\mathrm{CA}(N; 2, v)$, and sequentially adding columns, rejecting equivalent covering arrays after every step. Canonical augmentation [14, Sect. 4.2.3], [21] is used when extending representatives of covering arrays $\mathrm{CA}(N; k', v)$ to representatives of covering arrays $\mathrm{CA}(N; k' + 1, v)$; this part is described in detail in Section 3.1.2. Further, since our goal is to classify all $\mathrm{CA}(N; k, v)$ for certain k but not necessarily those that have a smaller number of columns, we can occasionally speed up the search by rejecting some partial arrays that cannot be extended to a full k-column covering array; the method is described in Section 3.1.3. When studying only uniform covering arrays, it

is easy to modify the algorithm to require uniformity.

3.1 Algorithm

In this section, we describe the algorithm for classifying all covering arrays $\mathrm{CA}(N; k+1, v)$, starting from a set of equivalence class representatives of covering arrays $\mathrm{CA}(N; k, v)$. To apply such an algorithm, we need a base case, which here is a classification of the covering arrays $\mathrm{CA}(N; 2, v)$. Since all v^2 pairs of symbols must occur in the two columns of those covering arrays, we may focus on the $N - v^2$ excess rows and just the equivalence issue. For the excess part in the first column, the symbol distributions are in one-to-one correspondence to the integer partitions of $N - v^2$ into at most v parts. In the second column we may take obvious symmetries into account to reduce the number of candidates considered. Finally, equivalent arrays are rejected.

3.1.1 Extending covering arrays

Consider a covering array C' obtained by adding a column to a covering array C. The symbols in the new column in C' induce a partition of the rows of C into covering arrays of strength 1. We call a subset of C that is a covering array of strength 1 a *cover* of C. If no proper subset of a cover of C is a cover of C, we call it a *minimal cover* of C. Each cover of C has one or more subsets that are minimal covers. For a cover D, we denote the lexicographically smallest subset that is a minimal cover by $\phi(D)$.

When extending covering arrays, we first determine $\mathcal{D}$, the set of all minimal covers of C. Then we find all sets $\{D_1, D_2, \ldots, D_v\}$ of v minimal covers that pack inside C, that is, $\biguplus_i D_i \subseteq C$. For each such set, we generate all full partitions $\{C_1, C_2, \ldots, C_v\}$ of C, where $D_i \subseteq C_i$ for all i, by adding the remaining rows in the sets D_i in all possible ways. To avoid repetition, we reject in the search all partitions for which $D_i \neq \phi(C_i)$. To get a covering array from an unlabeled partition, we map the symbols to the parts such that the resulting covering array is lexicographically smallest; this mapping from partitions to covering arrays is required in the sequel.

3.1.2 Isomorph rejection

Having generated all extensions of C up to permutation of symbols in the last column, canonical augmentation is used for isomorph rejection in two phases. The first phase rejects some arrays and ensures that two remaining arrays can be equivalent only if they were generated from the same C, and further that there is an automorphism of C that maps one onto another. The

second phase then accepts precisely one array from each equivalence class. Actually the two phases can be carried out in arbitrary order, and in our implementation Condition 3 below is checked first to help in validating the results, to be discussed later.

In the first phase, we use the v-tuple consisting of the counts of each symbol in that column sorted in descending order as an invariant of a column. For example, if a column contains three entries equal to 0, six entries equal to 1, and three entries equal to 2, then the invariant is $(6, 3, 3)$. A covering array C' passes the first phase of canonical augmentation if:

1. no other column has lexicographically smaller invariant than the last column, and

2. out of those columns with the same invariant as the last column, the last column is in the orbit that gets the smallest label in a canonical labeling by *nauty*.

Let μ be the largest multiplicity of a symbol in the column of C that has the smallest invariant. The first condition ensures that in a canonically augmented C', there is no symbol in the new column with multiplicity larger than μ. This allows us to remove from $\mathcal{D}$ all minimal covers with size larger than μ before the search begins and also not consider full partitions of C for which one part has size larger than μ.

For the second phase, we treat the array C' as a partition of C. Let c be an arbitrarily chosen row of C which has multiplicity 1 and is held fixed for the search of extensions of C. Let $\mathcal{C}$ be the orbit of C' under the action of $\mathrm{Aut}(C)$, and let $\chi(C', c) = \phi(A)$ where A is the part in C' that contains c. An array C' passes the second phase if

3. $\chi(C', c) \leq \chi(C'', c)$ for all $C'' \in \mathcal{C}$, and

4. C' is the smallest in the set $\{C'' \in \mathcal{C} : \chi(C'', c) = \chi(C', c)\}$, in terms of lexicographical ordering of the corresponding arrays.

Condition 3 allows us to reject from $\mathcal{D}$ all minimal covers D that contain c for which there is a $g \in \mathrm{Aut}(C)$ such that $c \in gD$ and $gD < D$. To this end, the row c is selected to be the one that maximizes the number of minimal covers that are rejected from $\mathcal{D}$.

3.1.3 A pruning condition

Let N and v be integers with $N < v(v+1)$. Let C be a $\mathrm{CA}(N; k, v)$ and let C' be a $\mathrm{CA}(N; k', v)$ that is obtained by adding $\delta = k' - k$ columns to C.

Since N is strictly less than $v^2 + v$, each of the last δ columns of C' contain at least one symbol of multiplicity v, each of which corresponds to a cover of size v in C. For each pair of columns and each symbol of multiplicity v, the two covers intersect in exactly one row (considering duplicate rows as separate elements). Thus C has a set of δ covers of size v which pairwise intersect in only one row.

If we are interested only in covering arrays $\mathrm{CA}(N; k', v)$ and not in covering arrays $\mathrm{CA}(N; k'', v)$ for any $k < k'' < k'$, we can restrict our search to the covering arrays $\mathrm{CA}(N; k, v)$ that satisfy this property. We gain further speedup by running the search for each possible way to fix the set of δ covers of size v that intersect in the desired way, as fixing the set allows rejecting many covers in $\mathcal{D}$ immediately.

3.1.4 Some implementation details

A core subroutine of the algorithm is that of finding subsets of $\mathcal{D}$ that pack inside C. This was implemented in two different ways, one using *Cliquer* [26] and one using *libexact* [15]. The simpler approach using *Cliquer* is faster in some cases, but in most cases, the approach using *libexact* is faster.

To use *Cliquer* we define G to be a graph with a vertex for every cover in $\mathcal{D}$ and an edge between two covers if their multiset sum is a subset of C. A packing corresponds to a clique of size v in G, but a clique may not be a valid packing if there are not enough duplicate rows in C. Further, when $N \geq v(v+1)$, two covers in a packing may be identical, so elements in $\mathcal{D}$ for which all rows have multiplicity greater than 1 in C must be represented with duplicated vertices in G.

The library *libexact* is used to find all solutions to a system of linear equations $Ax = b$ with $0 \leq x_j \leq u_j$ where A is a $(0,1)$-matrix. We set up the instance as follows. For each cover in $\mathcal{D}$, we have a variable whose value is the multiplicity of the cover in the packing. For each different row in C, we then have an inequality; namely, the row should occur in the packing at most as many times as it occurs in C. To encode this as an equality, we add a variable for each row that whose value is the slack in that inequality (that is, how many instances of the row in C are not covered by the packing). Further, to force a solution to have exactly v covers, we add a condition that the sum of variables corresponding to covers in $\mathcal{D}$ must be equal to v. The upper bounds of each variable are directly obtained from the equalities, as all variables are nonnegative.

We introduce further slack variables to account for conditions on the sizes of covers in the packing. These slack variables have no effect on the solutions but they speed up the search by identifying some branches that cannot lead

to a solution. For a valid packing, let M_v be the number of covers of size v, let M_{v+1} be the number of covers of size $v+1$, and let $M_{\geq v+2}$ be the number of covers of size at least $v+2$. We have

$$M_v \leq v, \tag{14}$$

$$vM_v + (v+1)M_{v+1} + (v+2)M_{\geq v+2} \leq N. \tag{15}$$

Here (15) is obtained by counting the rows in each cover. We define s_1 and s_2 to be the slack variables in (14) and (15), respectively, giving

$$s_1 + M_v = v, \tag{16}$$

$$s_1 + s_2 + M_{\geq v+2} = N - v^2, \tag{17}$$

where we used $M_v + M_{v+1} + M_{\geq v+2} = v$ to get (17). These equations can be directly implemented by writing M_v and $M_{\geq v+2}$ as sums of the variables corresponding to covers of size v or at least $v+2$, respectively. The upper bound of s_1 and s_2 in the *libexact* instance is set to $N - v^2$, which follows from (17).

3.2 Computational results

A classification was carried out for $v = 3, 4, 5, 6$ and values of N up to the computational limit. Specifically, $\mathrm{CA}(N; k, v)$ are classified for $10 \leq N \leq 14$ when $v = 3$, for $17 \leq N \leq 20$ when $v = 4$, for $26 \leq N \leq 29$ when $v = 5$, and for $37 \leq N \leq 40$ when $v = 6$. The full classification is performed for all possible values of k, except for the cases of $\mathrm{CA}(29; k, 5)$ and $\mathrm{CA}(40; k, 6)$, where some values of k were skipped using the method described in Section 3.1.3 to get to the cases of $\mathrm{CA}(29; 7, 5)$ and $\mathrm{CA}(40; 6, 6)$; the latter has 0 solutions so a $\mathrm{CA}(40; k, 6)$ exists exactly when $k \leq 5$.

Due to the computational time, we were unable to carry out a complete classification in the following cases: $\mathrm{CA}(21; k, 4)$ with $k \in \{3, \ldots, 8\}$; $\mathrm{CA}(30; k, 5)$ with $k \in \{3, \ldots, 8\}$; and $\mathrm{CA}(41; k, 6)$ with $k \in \{3, \ldots, 7\}$. For example, we predicted that classifying $CA(21, 7, 4)$ would take 130 core-years (4 months in the cluster).

In all these cases, the number of uniform arrays is also obtained. Finally, in the uniform cases, the classification of $\mathrm{UCA}(21; k, 4)$ and $\mathrm{UCA}(30; k, 5)$ is performed exhaustively and for $\mathrm{UCA}(41; k, 6)$ partially, skipping levels to get to $\mathrm{UCA}(41; 7, 6)$; the latter has 0 solutions so a $\mathrm{UCA}(41; k, 6)$ exists exactly when $k \leq 6$.

A complete table of results obtained for $\mathrm{CA}(N; k, v)$ and $\mathrm{UCA}(N; k, v)$ is given in Table 1. When δ is not given, all covering arrays and uniform

covering arrays were classified. When δ is given, the stated quantities are the numbers of covering arrays or uniform covering arrays obtained using the method in Section 3.1.3 with the given δ. These quantities are lower bounds for the numbers of all covering arrays and uniform covering arrays. In cases where the count of all covering arrays is not given, only uniform covering arrays were classified. *Cliquer* was used in the cases marked with †, and *libexact* was used in all other cases.

The times refer to a single logical core of an Intel Xeon E5 family processor with multi-threading enabled. Because a covering array occurs as a subset of a covering array with more rows, the classification results of smaller N could be obtained from the results of larger N; however, the running times are reported separately to give an idea of how the running time of the algorithm depends on N. The method to generate inequivalent 2-column arrays is not optimized and the time is not comparable to the other times so the time for $k = 2$ is not reported; in all cases the generation took less than 10 seconds.

Table 1: Detailed computational results

v	N	k	δ	# CA	# UCA	CPU time
3	10	2		1	1	
3	10	3		3	3	< 0.01 s
3	10	4		2	2	< 0.01 s
3	10	5		0	0	< 0.01 s
3	11	2		3	1	
3	11	3		20	9	0.01 s
3	11	4		27	8	0.02 s
3	11	5		3	3	0.01 s
3	11	6		0	0	< 0.01 s
3	12	2		7	1	
3	12	3		134	9	0.02 s
3	12	4		987	53	0.16 s
3	12	5		891	125	0.38 s
3	12	6		13	11	0.10 s
3	12	7		1	1	< 0.01 s
3	12	8		0	0	< 0.01 s
3	13	2		16	3	
3	13	3		937	151	0.09 s
3	13	4		53 523	12 747	6.0 s
3	13	5		739 845	302 524	144.5 s
3	13	6		752 165	506 680	940.8 s
3	13	7		24 934	22 539	600.9 s

Table 1: Detailed computational results (cont.)

v	N	k	δ	# CA	# UCA	CPU time	
3	13	8		5	5	11.4 s	
3	13	9		4	4	< 0.01 s	
3	13	10		0	0	< 0.01 s	
3	14	2		32	4		
3	14	3		5 973	476	0.51 s	
3	14	4		2 212 568	214 630	580.2 s	
3	14	5		325 046 812	43 473 308	29.9 h	
3	14	6		7 759 008 032	1 516 020 148	54.1 d	
3	14	7		18 844 482 204	5 827 703 442	446.8 d	
3	14	8		2 790 300 754	1 429 724 866	519.7 d	
3	14	9		17 068 936	12 725 845	43.4 d	
3	14	10		4 490	4 117	4.1 h	
3	14	11		0	0	3.3 s	
4	17	2		1	1		
4	17	3		6	6	0.03 s	
4	17	4		3	3	< 0.01 s	
4	17	5		4	4	< 0.01 s	
4	17	6		0	0	< 0.01 s	
4	18	2		3	1		
4	18	3		79	42	0.08 s	
4	18	4		79	31	0.13 s	
4	18	5		201	67	0.08 s	
4	18	6		0	0	0.04 s	
4	19	2		7	1		
4	19	3		1 365	191	0.51 s	
4	19	4		12 368	1 995	4.7 s	
4	19	5		74 113	1 495	16.1 s	
4	19	6		4	4	20.1 s	
4	19	7		0	0	< 0.01 s	
4	20	2		21	1		
4	20	3		30 334	183	16.0 s	
4	20	4		6 409 721	65 517	2265.4 s	
4	20	5		57 544 941	214 717	12.1 h	
4	20	6		25 760	745	32.6 h	
4	20	7		0	0		
4	21	2		47	3		
4	21	3			25 763	12.1 s	†
4	21	4			246 546 229	21.7 h	†

Table 1: Detailed computational results (cont.)

v	N	k	δ	# CA	# UCA	CPU time	
4	21	5			19 419 386 435	228.0 d	†
4	21	6			3 100 200 221	2326.4 d	†
4	21	7			1 005	135.4 d	†
4	21	8			0	2.9 s	†
5	26	2		1	1		
5	26	3		15	15	0.87 s	
5	26	4		3	3	0.07 s	
5	26	5		6	6	< 0.01 s	
5	26	6		6	6	0.01 s	
5	26	7		0	0	< 0.01 s	
5	27	2		3	1		
5	27	3		540	347	16.0 s	
5	27	4		385	193	7.2 s	
5	27	5		3 104	1 240	2.9 s	
5	27	6		11 603	3 463	14.1 s	
5	27	7		0	0	22.3 s	
5	28	2		7	1		
5	28	3		34 318	8 042	224.9 s	
5	28	4		263 321	70 992	894.5 s	
5	28	5		4 388 439	210 311	2874.9 s	
5	28	6		75 720 344	1 455 113	12.7 h	
5	28	7		0	0	54.5 h	
5	29	2		21	1		
5	29	3		2 243 097	69 891	5808.2 s	
5	29	4	3	148 843	19 884	44.3 h	
5	29	5	2	36 022	31 315	1565.3 s	
5	29	6	1	120 074	119 047	403.5 s	
5	29	7		281	258	412.5 s	
5	29	8		0	0	1.0 s	
5	30	2		54	1		
5	30	3			78 086	3947.2 s	
5	30	4			3 002 015 967	60.8 d	
5	30	5			5 501 626 305	645.8 d	
5	30	6			197 049 834	211.8 d	
5	30	7			18 857	165.0 h	
5	30	8			0	72.9 s	
6	37	2		1	1		
6	37	3		231	231	6.0 h	

Table 1: Detailed computational results (cont.)

v	N	k	δ	# CA	# UCA	CPU time	
6	37	4		13	13	6.2 s	
6	37	5		0	0	0.10 s	
6	38	2		3	1		
6	38	3		30 491	21 371	156.0 h	
6	38	4		8 865	6 215	1074.0 s	
6	38	5		0	0	143.7 s	
6	39	2		7	1		
6	39	3		5 128 096	1 644 791	82.1 d	
6	39	4		48 249 923	19 197 035	211.7 h	
6	39	5		289	158	248.1 h	
6	39	6		0	0	4.3 s	
6	40	2		21	1		
6	40	3		747 865 015	57 025 160	362.3 d	
6	40	4	2	471 192 731	85 773 975	19213.0 d	
6	40	5	1	388	128	192.1 d	
6	40	6		0	0	10.9 s	
6	41	2		54	1		
6	41	3			581 769 269	756.9 d	
6	41	4	3		1 771 354 037	3750.0 d	†
6	41	5	2		61 351	541.7 d	†
6	41	6	1		16	1154.5 s	†
6	41	7			0	0.33 s	

3.3 Double counting

To increase confidence in the computational results, we perform a consistency check of the results by double counting. After the search starting from $\mathrm{CA}(N; k, v)$ is performed, we count in two ways the total number of $\mathrm{CA}(N; k+1, v)$ that obey the restrictions used, that is, in some cases we count only uniform arrays and in some cases only arrays that have δ covers of size v that intersect pairwise in exactly one row.

The first way is to use the classification results to and the orbit-stabilizer theorem to obtain

$$\sum_{C'} \frac{(k+1)!v!^{k+1}}{|\mathrm{Aut}(C')|},$$

where $(k+1)!v!^{k+1}$ is the order of the group of symmetries in that case and the sum is taken over equivalence class representatives C' of that case.

The second way is to use numbers that were stored during the search. Consider first a modified search that starts from all k-column arrays instead of equivalence class representatives and considers all possible permutations of symbols in the last column for each partition. If the techniques for rejecting candidates of $\mathcal{D}$ in Section 3.1.2 would not be used, then every $(k+1)$-column covering array would appear exactly once when adding one more column.

If the additional condition on the largest multiplicity of a symbol in the last column is taken into account, then the proportion of $(k+1)$-column arrays equivalent to C' that enter the isomorph rejection phase is the proportion of columns in C' for which the largest multiplicity of a symbol is smallest, denoted by $\alpha(C')$. Further, in the search starting from a fixed k-column array C, let $\beta(C, C')$ be the proportion of all partitions equivalent to C' that pass the check 3 in Section 3.1.2; this can be obtained at the stage in the search when all partitions of C equivalent to C' are considered. The total count of $(k+1)$-column arrays would now be

$$\sum_{C,C'} \frac{1}{\alpha(C')\beta(C,C')},$$

where the sum is taken over all k-column arrays C and all C' that are extensions of C and pass the check for condition 3. The remaining techniques for rejecting covers in $\mathcal{D}$ described in Section 3.1.2 do not reject any candidates that satisfy condition 3, so including them in the search does not change this count.

In the modified search, arrays C' which differ only by a permutation of symbols in the last column contribute the same amount in the sum, and the searches starting from two equivalent C contribute the same amount to the sum. Let $S(C')$ be the number of ways to assign the symbols to the last column (this equals $v!$ if no two parts in the corresponding partition of C' are equal). Further, the size of the equivalence class of C is $k!v!^k/|\mathrm{Aut}(C)|$. In the actual search, the count is then obtained as

$$\sum_{C,C'} \frac{k!v!^k}{|\mathrm{Aut}(C)|} S(C') \frac{1}{\alpha(C')\beta(C,C')},$$

where the sum is taken over the k-column arrays C that are used in the search, and all C' that are extensions of C and pass the check for condition 3.

v	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	9	${}^{j}11^{i}$	${}^{l}12^{c}$	${}^{c}12^{k}$	${}^{a}13^{c}$	${}^{c}13^{k}$	${}^{a}14^{h}$
4	16	16	${}^{l}19^{k}$	${}^{a}21^{k}$	${}^{c}21$–22^{c}	${}^{c}21$–22^{f}	${}^{c}21$–24^{h}
5	25	25	25	${}^{l}29^{k}$	${}^{b}30$–33^{h}	${}^{c}30$–35^{h}	${}^{c}30$–36^{e}
6	${}^{m}37^{k}$	${}^{a}39^{d}$	${}^{b}41^{h}$	${}^{c}41$–42^{c}	${}^{c}41$–42^{h}	${}^{c}41$–46^{g}	${}^{c}41$–48^{n}

Unmarked entries: orthogonal arrays. Captions: a) This paper, preliminarily announced in [28], b) This paper, c) $\mathrm{CAN}(k,v) \leq \mathrm{CAN}(k+1,v)$, d) [3], e) L. Rouse-Lamarre, reported in [7], f) [20], g) [24], h) [27], i) [29], j) Applegate, reported in [35], k) [39], l) [40], m) [41], n) [13]

Table 2: Values of $\mathrm{CAN}(k,v)$ for $4 \leq k \leq 10$ and $3 \leq v \leq 6$

4 Discussion of results

A summary of the current knowledge of the sizes of optimal coverings array for small k and v is given in Table 2. In the table there are captions for all bounds, except those that follow from orthogonal arrays: $\mathrm{CAN}(k,v) = v^2$ when v is a prime power and $k \leq v+1$. Some of the lower bounds attributed to the current work were obtained about two decades before this paper appears in print. Those bounds, which were announced at a conference in 2000 [28], are given a caption of their own to clarify priority issues. Some of those results have later been rediscovered [2, 7, 42, 43]. Additionally our computer search established that a $\mathrm{CA}(14;11,3)$ does not exist. Nurmela found a $\mathrm{CA}(15;20,3)$ [27] so we additionally know that $\mathrm{CAN}(k,3) = 15$ for $11 \leq k \leq 20$.

In the process of preparing this paper we noticed that the bound sources listed in Table 2 of [27] are not the same as those in Table 1 but this difference is not articulated in that article. To the best of our knowledge in Table 2 of [27], b refers to [37], c is [27] and d is [38]. On page 149 of [27], "giving the bounds marked with d in the tables" should read "giving the bounds marked with b in Table 1 and c in Table 2".

In every case in which we determined the size of an optimal covering array by construction, we also determined that there exists a uniform covering array of the same size. These results continue to support the conjecture that optimal covering arrays can be found amongst the uniform covering arrays.

For the parameters $\mathrm{CA}(11;5,3)$, $\mathrm{CA}(12;7,3)$, $\mathrm{CA}(13;8,3)$, $\mathrm{CA}(13;9,3)$, $\mathrm{CA}(19;6,4)$ and $\mathrm{CA}(37;4,6)$ every optimal array is also uniform. However for the optimal parameters, $\mathrm{CA}(12;6,3)$, $\mathrm{CA}(14;10,3)$, $\mathrm{CA}(29;7,5)$ and $\mathrm{CA}(39;5,6)$ both uniform and non-uniform examples exist. Finally, for the

v	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
3	**9**	**11**	**12**	**12**	**13**	**13**	**14**
4	**16**	**16**	**19**	**21**	22	22	22
5	**25**	**25**	**25**	**29**	31	32	32
6	**37**	**39**	**41**	42	42	44	45

Table 3: Lower bounds on UCAN(k, v) for $4 \leq k \leq 10$ and $3 \leq v \leq 6$

optimal parameters CA$(21; 7, 4)$ and CA$(41; 6, 6)$ we know that uniform arrays exist, but we do not know if non-uniform examples also exist.

For the four optimal parameter sets where both uniform and non-uniform arrays exist, CA$(12; 6, 3)$, CA$(14; 10, 3)$, CA$(29; 7, 5)$ and CA$(39; 5, 6)$, the percentages of non-isomorphic arrays that are uniform are 84.61, 91.69, 91.81 and 54.67 respectively. Combined with the optimal parameter sets where every array is uniform we can see some provisional trends. The smallest optimal parameter set for which non-uniform arrays exist is CA$(12; 6, 3)$, for which N is the smallest for the given k, but k is not the maximal possible given N. We guess that non-uniform arrays will be more abundant when the k is not maximal for a given N. The second potential trend is that for a fixed v, as N and k increase there are likely to be more non-uniform optimal arrays. These are only limited observations from few data.

Table 3 shows the current state of knowledge for uniform covering arrays. All entries are lower bounds, bold entries show arrays known to exist, and underlined entries indicate a lower bound matching Theorem 1. The lower bounds from Theorem 1 meet six known uniform covering arrays.

A lower bound in Table 2 that is smaller than the corresponding lower bound in Table 3 indicates a candidate for a covering array that would refute Conjecture 1.

Conjecture 1 and Corollary 2 would imply that CAN$(v+2, v) \geq v^2+v-1$, which has also been conjectured in [37]. When v is a prime power, this would mean that CAN$(v+2, v) -$ CAN$(v+1, v) \geq v-1$, which is a very large jump for only adding a single column. In the case of $v = 6$, from Table 2 we can see that the value of CAN$(v+1, v)$ may be close to the value of CAN$(v+2, v)$ when v is not a prime power.

One exciting possibility is that CAN$(8, 6)$ could be 41, meeting the bound from Theorem 1; CAN$(8, 6)$ is no more than 42. This suggests that the influence of the prime power status of v disappears very rapidly as k increases past $v + 1$. However, none of the UCA$(41; 6, 6)$ covering arrays can be extended

to a CA$(41; 8, 6)$, so if they exist, then no subarray with six columns can be uniform. This implies that at least three columns of a possible CA$(41; 8, 6)$ must be non-uniform. Since there are six non-uniform partitions of 41 into six parts of size at least 6 there are 57 different partition patterns if only three columns are non-uniform. If more columns are non-uniform, the number of cases increases. This indicates that exploiting this structure in an exhaustive search may not be efficient. Exploiting it with a metaheuristic search could be an option.

Conjecture 1 predicts that for every covering array, there a uniform covering array with the same parameters. We have seen examples of optimal covering arrays which are not uniform, so we know that not every optimal covering array is uniform. But we can ask for which parameters are all the optimal covering arrays uniform? In this paper we found many examples, but our examples are in cases where the number of rows is relatively small. When v is a prime power it is possible to construct a $CA(v^2 + i(v^2 - v), v^i(v + 1), v)$ for any i using a recursive construction and starting with an orthogonal array(see for example [37]). We suspect that these parameters could be good candidates for having every optimal covering arrays be a uniform covering array.

A significant result from our work is that the number of known optimal covering arrays for $v > 2$ and $N > v^2$ is now 21 whereas before it was eleven. Additionally the UCA$(21; 7, 4)$ meet the bound from Theorem 1. This is the first example of tightness, and the implied structure, when $k > v + 2$. The classification results from our searches are available at [18].

In this paper we only consider strength-2 covering arrays. Many of the questions addressed in the paper may be interesting for higher strength covering arrays. The definition of "uniform" applies to covering arrays of any size and it is interesting to ask if it is always possible to find an optimal covering array, of any strength, that is also uniform. Extending Theorem 1 to strength t would require counting pairs of rows which agree in at most $t - 1$ positions and would be an interesting investigation. For strength $t > 2$, extending an array with an additional column requires determining all of the strength $t - 1$ subarrays which is more computationally demanding as t increase. The use of nauty to compute symmetry groups depends only on the sizes of the arrays and is not inherently more complicated as the strength increases. For classification and fully understanding the structure of optimal arrays, we do not see better options than exhaustive search.

Acknowledgment

The authors also wish to thank the referees for useful comments that helped improve this article.

References

[1] Y. Akhtar, S. Maity, and R. C. Chandrasekharan, Covering arrays of strength four and software testing, in: R. N. Mohapatra, D. R. Chowdhury, and D. Giri (Eds.), Mathematics and Computing, Springer Proc. Math. Stat. 139, Springer, New Delhi, 2015, pp. 391–398.

[2] M.B. Cohen, personal communication 2014.

[3] M.B. Cohen, Designing test suites for software interaction testing, Ph.D. Thesis, The University of Auckland, 2004.

[4] C. J. Colbourn, Combinatorial aspects of covering arrays, Le Matematiche (Catania), **59** (2006), 125–172.

[5] C. J. Colbourn, Augmentation of covering arrays of strength two, Graphs Combin. **31** (2015), 2137–2147.

[6] C. J. Colbourn, Suitable permutations, binary covering arrays, and Paley matrices, Springer Proc. Math. Stat. **133** (2015) 29–42.

[7] C. J. Colbourn, G. Kéri, P. P. R. Soriano, and J.-C. Schlage-Puchta, Covering and radius-covering arrays: constructions and classification, Discrete Appl. Math. **158** (2010), 1158–1180.

[8] C. J. Colbourn, K. Sarkar, and E. Lanus, Asymptotic and constructive methods for covering perfect hash families and covering arrays, Des. Codes Cryptogr. **86** (2018), 907–937.

[9] P. Danziger, E. Mendelsohn, L. Moura and B. Stevens, Covering arrays avoiding forbidden edges, Theoret. Comput. Sci. **410** (2009), 5403–5414.

[10] N. Francetić and B. Stevens, Asymptotic size of covering arrays: an application of entropy compression, J. Combin. Des. **25** (2017), 243–257.

[11] A. Hartman, Software and hardware testing using combinatorial covering suites, in: M. C. Golumbic and I. B.-A. Hartman (Eds.), Graph Theory, Combinatorics and Algorithms, Springer, New York, 2005, pp. 237–266.

[12] Brahim Hnich, Steven D, Prestwich, Evgeny Selensky and Barbara M. Smith, Constraint models for the covering test problem, Constraints. **11** (2006), 199–219.

[13] I. Izquierdo-Marquez, J. Torres-Jimenez and H. Avila-George, New Upper Bounds for Pairwise Senary Test-Suites, submitted to The International Arab Journal of Information Technology, 2018.

[14] P. Kaski and P. R. J. Östergård, Classification Algorithms for Codes and Designs, Springer, Berlin, 2006.

[15] P. Kaski and O. Pottonen, libexact user's guide, Version 1.0, Helsinki Institute for Information Technology HIIT, Helsinki, 2008.

[16] G. O. H. Katona, Two applications (for search theory and truth functions) of Sperner type theorems, Period. Math. Hungar. 3 (1973), 19–26.

[17] D. J. Kleitman and J. Spencer, Families of k-independent sets, Discrete Math. **6** (1973), 255–262.

[18] J. I. Kokkala, K. Meagher, R. Naserasr, K. J. Nurmela, P. R. J. Östergård, and B. Stevens, Dataset for On the structure of small strength-2 covering arrays [Dataset]. Zenodo. https://doi.org/10.5281/zenodo.1476059 (October 31, 2018).

[19] J. Lawrence, R. N. Kacker, Y. Lei, D. R. Kuhn, and M. Forbes, A survey of binary covering arrays, Electron. J. Combin. **18** (2011), P84.

[20] J. R. Lobb, C. J. Colbourn, P. Danziger, B. Stevens, and J. Torres-Jimenez, Cover starters for covering arrays of strength two, Discrete Math. **312** (2012), 943–956.

[21] B. D. McKay, Isomorph-free exhaustive generation, J. Algorithms **26** (1998), 306–324.

[22] B. D. McKay and A. Piperno, Practical graph isomorphism, II, J. Symbolic Comput. **60** (2014), 94–112.

[23] K. Meagher and B. Stevens, Covering arrays on graphs, J. Combin. Theory Ser. B **95** (2005), 134–151.

[24] K. Meagher and B. Stevens, Group construction of covering arrays, J. Combin. Des. **13** (2005), 70–77.

[25] Peyman Nayeri, Charles J. Colbourn, and Goran Konjevod, Randomized post-optimization of covering arrays, European J. Combin. **34** (2013), 91–103.

[26] S. Niskanen, P. R. J. Östergård, Cliquer User's Guide, Version 1.0, Tech. Rep. T48, Communications Laboratory, Helsinki University of Technology, Espoo, 2003.

[27] K. J. Nurmela, Upper bounds for covering arrays by tabu search, Discrete Appl. Math. **138** (2004), 143–152.

[28] K. J. Nurmela and P. R. J. Östergård, Lower bounds on 2-covering arrays by exhaustive search, presented at the *25th Australasian Conference on Combinatorial Mathematics and Combinatorial Computing* (Christchurch, New Zealand, December 4–8, 2000).

[29] P. R. J. Östergård, Constructions of mixed covering codes, Research Report A18, Digital Systems Laboratory, Helsinki University of Technology, Espoo, 1991.

[30] P. R. J. Östergård, Disproof of a conjecture on the existence of balanced optimal covering codes, IEEE Trans. Inform. Theory **49** (2003), 487–488.

[31] P. R. J. Östergård, On optimal binary codes with unbalanced coordinates, Appl. Algebra Engrg. Comm. Comput. **24** (2013), 197–200.

[32] P. R. J. Östergård, T. Baicheva, E. Kolev, Optimal binary one-error-correcting codes of length 10 have 72 codewords, IEEE Trans. Inform. Theory **45** (1999), 1229–1231.

[33] K. Sarkar and C. J Colbourn, Upper bounds on the size of covering arrays, SIAM J. Discrete Math. **31** (2017), 1277–1293.

[34] K. Sarkar, C. J. Colbourn, A. de Bonis, and U. Vaccaro, Partial covering arrays: algorithms and asymptotics, in: V. Mäkinen, S. J. Puglisi and L. Salmela (Eds.), Combinatorial Algorithms, LNCS 9843, Springer, Cham, 2016, pp. 437–448,

[35] N. J. A. Sloane, Covering arrays and intersecting codes, J. Combin. Des. **1** (1993), 51–63.

[36] B. Stevens, Non-uniform covering array with symmetric forbidden edge constraints, arXiv e-prints (2019) arXiv:1901.02479, available at `https://arxiv.org/abs/1901.02479`. Submitted to Australas. J. Combin.

[37] B. Stevens, Transversal Covers and Packings, Ph.D. Thesis, University of Toronto, Toronto, 1998.

[38] B. Stevens, A. Ling and E. Mendelsohn, A direct construction of transversal covers using group divisible designs, Ars Combin. **63** (2002) pp. 145–159.

[39] B. Stevens and E. Mendelsohn, New recursive methods for transversal covers, J. Combin. Des. **7** (1999) pp. 185–203.

[40] B. Stevens, L. Moura, and E. Mendelsohn, Lower bounds for transversal covers, Des. Codes Cryptogr. **15** (1999), pp. 279–299.

[41] G. Tarry, Le Problèm des 36 Officiers, C. R. Assoc. Fr. Av. Sci. **29**(2) (1900), pp. 170–203.

[42] J. Torres-Jimenez, personal communication 2016.

[43] J. Torres-Jimenez and I. Izquierdo-Marquez, Construction of non-isomorphic covering arrays, Discrete Math. Algorithms Appl. **8** (2016), 1650033.

[44] J. Torres-Jimenez and E. Rodriguez-Tello, New bounds for binary covering arrays using simulated annealing, Inform. Sci. **185** (2012) pp. 137–152.

[45] G. Tzanakis, L. Moura, D. Panario, and B. Stevens, Covering arrays from m-sequences and character sums, Des. Codes Cryptogr. **85** (2017), 437–456.

[46] R. Yuan, Z. Koch, and A. Godbole, Covering array bounds using analytical techniques, Congr. Numer. **222** (2014), 65–73.