A Survey of Binary Covering Arrays

Jim Lawrence

George Mason University, Fairfax, VA 22030
lawrence@gmu.edu

Raghu N. Kacker

National Institute of Standards and Technology, Gaithersburg, MD 20899
raghu.kacker@nist.gov

Yu Lei

University of Texas at Arlington, Arlington, TX 76019
ylei@cse.uta.edu

D. Richard Kuhn

National Institute of Standards and Technology, Gaithersburg, MD 20899
d.kuhn@nist.gov

Michael Forbes

Massachusetts Institute of Technology, Cambridge, MA 02139
miforbes@mit.edu

Submitted: Jun 21, 2010; Accepted: Mar 31, 2011; Published: Apr 7, 2011
Mathematics Subject Classifications: 05B20, 05B30

Abstract

Binary covering arrays of strength t are 0–1 matrices having the property that for each t columns and each of the possible 2^t sequences of t 0's and 1's, there exists a row having that sequence in that set of t columns. Covering arrays are an important tool in certain applications, for example, in software testing. In these applications, the number of columns of the matrix is dictated by the application, and it is desirable to have a covering array with a small number of rows. Here we survey some of what is known about the existence of binary covering arrays and methods of producing them, including both explicit constructions and search techniques.

1. Introduction.

An $n \times k$, $(v_1, \ldots, v_k)$-*valued covering array of strength* t, where n, t, and k are integers satisfying $n \geq 1$ and $1 \leq t \leq k$, and $(v_1, \ldots, v_k)$ is a vector of k integers $v_j \geq 2$, is a matrix of size $n \times k$ such that

- entries in column j come from a set V_j of "parameter values" of cardinality v_j, and
- each $n \times t$ submatrix having columns indexed by elements of a set $\Lambda \subseteq \{1, \ldots, k\}$, where $|\Lambda| = t$, contains all of the different $\prod_{j \in \Lambda} v_j$ possible rows that can be formed by choosing the entry with index $j \in \Lambda$ from the j-th set of parameter values.

Suppose $\Lambda = \{j_1, \ldots, j_t\} \subseteq \{1, \ldots, k\}$, where $j_1 < j_2 < \ldots < j_t$. We call the pair (Λ, x), $x = (x_{j_1}, \ldots, x_{j_t})$ being a vector whose entries are indexed by Λ, a *t-tuple.* If $y = (y_1, \ldots, y_k)$ is a vector of length k, we say that y *covers* the t-tuple (Λ, x) provided that $y_j = x_j$ when $j \in \Lambda$. A covering array has the property that each t-tuple whose entries come from the corresponding parameter sets is covered by some row of the array.

Although much work has been done concerning the more general case, this paper will be almost exclusively concerned with the 2-*valued (binary)* case, in which $v_j = 2$ for $1 \leq j \leq k$. A $(v_1, \ldots, v_k)$-valued covering array, where the v_j's are all equal to v, $v_j = v$ for $1 \leq j \leq k$, will be termed a v-*valued* covering array.

A covering array of strength $t = 2$ is sometimes called a *pairwise covering array.* Such arrays have proven useful in a variety of settings. The use of such arrays in applications is possible thanks to a variety of software for their construction. When a higher strength is desired, the problem of construction of the covering arrays becomes more difficult, and the development of software for this is at a less advanced stage.

There are many settings in which covering arrays may be useful. An early use of covering arrays (framed in terms of a "piercing" set of vertices of the cube) was that of Nečiporuk [65], where a result was established and used to bound the complexity of certain Boolean gating circuits. We describe a couple of other applications in the following subsection. Additional applications may be found in [7], [11], [16], [18], [33], [53], [54], [79], [80], and [86].

Some applications for covering arrays. Covering arrays are useful in a method of software or hardware testing, proposed in [25] and elsewhere. Suppose a certain computer program requires as input the values of k parameters, where the j-th parameter has v_j possible values $(1 \leq j \leq k)$. An $n \times k$, $(v_1, \ldots, v_k)$-valued covering array of strength t can be used to provide n tests, one for each row of the array, such that, for each choice of t of the parameters and any choice of values for those t parameters, at least one of the tests provides those values for that set of t parameters. Certainly, if not all possible inputs are tested, then the program may contain errors that are not detected; however there is evidence to suggest that in most cases errors are the result of interactions among a small number of the parameter values; see Kuhn, Reilly [53] and Kuhn, Wallace, Gallo [54]. Furthermore, testing all of the possible parameter settings is often not feasible because of the great number of them. For example, in the binary case in which $v_1 = \ldots = v_k = 2$, the total number of possible parameter settings is $n = 2^k$. If instead we choose, for each possible set of t parameters and each of the 2^t possible settings of these t parameters, a test providing these values, we will have $n = 2^t\binom{k}{t}$

tests, which is much smaller than 2^k when k is large compared to t. It is nevertheless much larger than the smallest number n of tests that will do, which is (for t fixed and large k) on the order of a constant multiple of $\log(k)$. (See Section 4.3.)

We are led to consider the following two closely related questions. Given k, t, and $(v_1, \ldots, v_k)$, how can an $n \times k$ $(v_1, \ldots, v_k)$-valued covering array of strength t be constructed, if it is desired that n be "not too large" and that the computational difficulty be "not too great"? Given k, t, and $(v_1, \ldots, v_k)$, what is the smallest value of n such that there exists such a covering array? These two questions may seem to be almost identical; but there is actually quite a big difference. We will see in Section 5 that many algorithms exist and indeed have been implemented to answer the first question to a degree that is often adequate; but, as we will see in Section 3, the answer for the second is known only in a very few cases.

Covering arrays and their relatives have also found use in a number of other applications. One of these, specific to binary covering arrays, concerns a problem having to do with *hypercube computers* (see Becker and Simon [5], Graham, Harary, Livingston, and Stout [41]). In such a computer, for some k, there are 2^k processors corresponding to the 2^k vertices of a k-dimensional cube, which are connected by links according to the pattern dictated by the edges of the cube. The analysis of the question, "What if some of the processors fail?" leads to problems concerning covering arrays. In particular, the question, "If n processors fail, what is the largest dimension of a (cubical) face of the k-cube that is nevertheless guaranteed to have all processors functioning?" leads to the same mathematical question concerning the minimum possible value of n as that for the software testing application. Given that the answer is less than a positive integer s, there is a set of n vertices of the k-cube that has nonempty intersection with each s-dimensional face of the cube, and these vertices form the rows of an $n \times k$ covering array of strength $t = k - s$. For given k and t (and s), we want to know the least possible value of n.

There is a big difference in the two examples with respect to the way in which covering arrays are involved. In the software-testing example, a small number of tests, that is, of rows of the matrix, is preferable, since this means that the testing can be done more quickly. In the hypercube computer example, one wants assurance that a larger number is necessary, since this is an indication that the computer will be less susceptible to failure.

It is possible for a (somewhat) happy ending in both cases. In the software-testing application, the parameter t may be considered to be fixed, and in this case the smallest n is comparable to $\log(k)$. If, in the hypercube computer example, one is happy with the assurance that some s-dimensional face will remain operational, then with $s = k - t$ constant, n increases exponentially as 2^k.

For a quite different application, Hartman [44] has applied covering arrays to a problem considered by Lim and Alpern [57] and Gal [39], namely, the problem of "blind dyslectic synchronized robots on a line." In this problem, there are k robots, $R_1, \ldots, R_k$, on a line, initially placed (in some order) at positions 1, 2, ..., k. Blindness here means that a robot can sense the presence of another only by touch; dyslectic means that the robots do not have a common sense of right and left on the line. Each robot can move on the line until it meets another. One wishes to determine the minimum over all possible strategies of the maximum over all possible starting permutations of the robots, of the time by which they can arrive at the same point. Hartman shows that this value is $\lceil k/2 \rceil + \lceil \log_2 k \rceil + 1$, by making use of a bound described in 3.2.1, below.

For given small values of $k = s + t$ and t, Table 1 records the smallest number of rows (n, denoted by $\mathrm{CAN}(k,t)$) in a binary covering array having those parameters, when this is known. When a range is given, the numbers represent lower and upper bounds on the smallest number of rows. This table is included to aid in the exposition. Much more thoroughgoing tables may be found at the website of Colbourn [28], from which many of the upper bounds in the table were derived; and we note that the upper bounds for the intervals in the lower right corner of this table represent recent results and will probably change again soon!

The tables of [28] do not include actual covering arrays (and we have not independently verified in all cases that covering arrays of the indicated sizes exist). For an extensive collection of actual covering arrays with relatively few rows, visit the webpage [87], and follow the link to the covering array library. Also, covering arrays are available at the website of Nurmela [67], and covering arrays can be downloaded, by request, from the website maintained by Torres-Jimenez [81]. The website of Torres-Jimenez includes, in particular, covering arrays yielding all of the upper bounds in Table 1.

$s\backslash t$	1	2	3	4	5	6
0	2	4	8	16	32	64
1	2	4	8	16	32	64
2	2	5	10	21	42	85
3	2	6	12	24	48–52	96–108
4	2	6	12	24	48–54	96–116
5	2	6	12	24	48–56	96–118
6	2	6	12	24	48–64	96–128
7	2	6	12	24	48–64	96–128
8	2	6	12	24	48–64	96–128
9	2	7	15	30–32	60–64	120–128
10	2	7	15–16	30–35	60–79	120–179

Table 1. Values of $\mathrm{CAN}(s+t,t)$.

In Section 2 we describe several ways of "looking at" covering arrays, including alternative terminology and definitions that have been used. Section 3 gives the values of CAN that are known exactly, including descriptions of the arguments and covering arrays that have been used to establish the lower and (equal) upper bounds. Section 4 describes the lower and upper bounds on values of CAN. Section 5 describes methods used in construction of covering arrays. Section 6 briefly addresses issues of computational complexity for some problems related to the existence and construction of covering arrays.

We do not survey the extensive collection of software available for constructing covering arrays. For such a survey see the paper by Grindal, Offutt, and Andler [43].

2. Various Formulations.

The phrase "covering array" was used in the paper by Sloane [75], to contrast these arrays with "orthogonal arrays," and this terminology is now common. An *orthogonal array* having parameters n, t, k, v, and $\lambda \geq 1$ is an $n \times k$ matrix with entries from a set of v elements, having the property that given any set of t columns each of the possible assignments of values to the corresponding parameters occurs the same number λ times. Here, $\lambda \geq 1$. When such an array exists, it is a v-valued $n \times k$ covering array of strength t; and when additionally $\lambda = 1$, it is easily seen to be a covering array for which n is minimized. In the binary case this fact is of little value, since such orthogonal arrays exist only in case $k = t$ or $t + 1$. However, orthogonal arrays for which λ is larger do have some relevance in the binary case; see section 4.1. Also, some methods exist to indirectly construct binary covering arrays by utilizing orthogonal arrays for which the parameter v is larger than 2.

There are several operations that can be performed on a covering array that yield covering arrays. The rows may be permuted. The columns may be permuted; in this case, a $(v_1, \ldots, v_k)$-valued covering array yields a covering array, but with the indices of the v_i's similarly permuted – no change, when the v_i's are equal. In any column, the values may be permuted; for example, in the binary case, in any column, the 0's and 1's may be switched. These operations determine equivalence relations on collections of covering arrays.

Covering arrays can be viewed in different ways. As described above, the columns may be given labels associated with parameters, the entries in each column being among the values that the corresponding parameter can have, and each row corresponds to a setting of the parameters for a "test." Alternatively, if the rows are labeled by the elements of a set, each column determines a partition of the set into those row-labels for which the entries in the column are the same. The resulting family of partitions is then "t-wise qualitatively independent"; this is covered in more detail in section 2.2, in the binary case. Also, in the binary case, each row of the covering array C may be viewed as a vertex of the k-dimensional cube, $[0, 1]^k$. Then the rows form a set of vertices of the cube having the property that, under orthogonal projection to any t-dimensional face F of the cube, the image of the set equals the set of vertices of F. This can be generalized to arbitrary sets of values, and the characteristic property is often called "t-surjectivity." See section 2.1. Again, in the binary case, viewing the set of rows as a set of vertices of the cube $[0, 1]^k$, it is not hard to see that each face F' of the cube having dimension $k - t$ must contain at least one of these vertices; so the notion of a "$(k - t)$-face transversal" is yet another equivalent to that of a covering array. See section 2.1.

2.1. The notion of a t-surjective mapping; transversals of s-faces. Let

$$X = \prod_{i=1}^{k} V_i,$$

where, for $1 \leq j \leq k$, V_j is the set of v_j possible values for the j-th parameter. Given a nonempty set $\Lambda = \{i_1, \ldots, i_t\} \subseteq \{1, 2, \ldots, k\}$, where $1 \leq i_1 < i_2 < \cdots < i_t \leq k$, let

$$X_\Lambda = \prod_{i \in \Lambda} V_i$$

so that in particular $X_{[k]} = X$, and let $\pi_\Lambda : X \to X_\Lambda$ be the projection

$$\pi_\Lambda\big((x_1, \ldots, x_k)\big) = (x_{i_1}, \ldots, x_{i_t}).$$

If T is a subset of X, we call the set T a *t-surjective set* if, for each set $\Lambda \subseteq [k]$ having cardinality t, the image $\pi_\Lambda(T)$ equals X_Λ; that is, the restriction of the function π_Λ to T, $\pi_\Lambda|_T : T \to X_\Lambda$, is surjective. The covering arrays of strength t are precisely the matrices whose rows form the elements of a t-surjective set.

Covering arrays under the name *t-surjective arrays* have been studied in [20] in the binary case, and in [19], and elsewhere.

In the binary case, we take the set of parameter values to be $\{0, 1\}$. Then $X = \{0, 1\}^k$, and $X_\Lambda = \{0, 1\}^\Lambda$, for $\Lambda \subseteq [k]$. We may consider the set $\{0, 1\}^k$ as a subset of the real vector space R^k. It is the set of vertices of the k-dimensional cube $[0, 1]^k$. It is easy to describe the t-surjective sets in geometrical terms, making use of this cube. Given an integer s such that $0 \leq s \leq k$, we will call a set T of vertices of the cube $[0, 1]^k$ an *s-face transversal* if, for each s-dimensional face F of the cube, $F \cap T \neq \emptyset$. The nonempty faces of this cube are precisely the inverse images $\pi_\Lambda^{-1}(v)$, where $\Lambda \subseteq \{1, \ldots, k\}$ and $v \in \{0, 1\}^\Lambda$ is a vertex of $[0, 1]^\Lambda$. The dimension of the face $\pi_\Lambda^{-1}(v)$ is $k - |\Lambda|$. If T is a t-surjective subset of $\{0, 1\}^k$, then, for Λ of cardinality t, since the restriction of the mapping π_Λ to T is surjective, for each $v \in \{0, 1\}^\Lambda$, T must contain at least one element of $\pi_\Lambda^{-1}(v)$; that is, T must have a point in common with the face $\pi_\Lambda^{-1}(v)$, which is an arbitrary face of dimension $k - t$. Therefore T is a transversal of the set of $(k - t)$-dimensional faces of the cube. Let s denote the difference $k - t$, so that $s + t = k$. A set T of elements of $\{0, 1\}^k$ is t-surjective if and only if it is an s-face transversal.

Johnson and Entringer, in [46], have considered the equivalent question of the maximum cardinality of subsets of the k-cube that do not contain the set of vertices of any s-face. Such a set is the complement (with respect to the set of the vertices of the k-cube) of an s-face transversal. Therefore, if the largest size of such a set is μ, then $\mathrm{CAN}(k, k - s) = 2^k - \mu$, so the problem of determining $\mathrm{CAN}(k, t)$ is equivalent to the problem of determining μ, where $s = k - t$.

2.2. Qualitative independence. Let S denote a set having n elements. Two subsets A and B of S are termed *qualitatively independent* if none of the sets $A \cap B$, $\bar{A} \cap B$, $A \cap \bar{B}$, and $\bar{A} \cap \bar{B}$ (where $\bar{A}$ and $\bar{B}$ denote the complements of A and B in S) is empty. Intuitively, this means that knowledge of whether or not an element x is in A does not indicate whether or not it is in B (and vice-versa). This notion was introduced by Marczewski [59], and it is described in Rényi's book [68]. It extends in the obvious fashion to collections of more than two sets: A collection of sets $A_1, \dots, A_t$ is qualitatively independent if each of the 2^t intersections $X_1 \cap X_2 \cap \cdots \cap X_t$, where each X_i is either A_i or $\bar{A}_i$, is nonempty.

Let $A_1, \dots, A_k$ be subsets of $[n]$ such that each family of t of these sets is qualitatively independent. Such a family of sets is called *t-independent.* Let C denote the $n \times k$ matrix for which the (i, j)-th entry is 1 if $i \in A_j$, and 0 otherwise. The j-th column of C is then the *indicator vector* of A_j. Since each t of the sets form a qualitatively independent family, each t columns of C form a matrix in which each of the 2^t possible vectors of 0's and 1's occurs at least once; that is, C is a binary covering array of strength t.

Problem 1. Let t and k for which $1 \leq t \leq k$ be given. What is the least value of n such that there exists a binary $n \times k$ covering array of strength t?

This minimum value is $\mathrm{CAN}(k, t)$. We also denote the set of binary $n \times k$ covering arrays of strength t by $\mathbf{CA}(n, k, t)$; then $\mathrm{CAN}(k, t)$ is the smallest number n such that $\mathbf{CA}(n, k, t) = \emptyset$.

Problem 1′. Given n and t, what is the largest number k such that there exist k subsets of $[n]$, each t of which are qualitatively independent?

We denote this maximum by $\mathrm{CAK}(n, t)$. Clearly CAK and CAN are related:

$$\mathrm{CAN}(k, t) = \min\{n : \mathrm{CAK}(n, t) \geq k\}$$

and

$$\mathrm{CAK}(n, t) = \max\{k : \mathrm{CAN}(k, t) \leq n\}.$$

Therefore, if either CAK or CAN is known for all values of the parameters, then the values for the other can be determined.

Covering arrays which are not necessarily binary can be studied similarly, using the notion of qualitative independence of families of partitions, rather than of subsets, of a set. This notion is also described in [68]. Covering arrays are then families of partitions, each t of which are qualitatively independent.

3. Known Values of $\mathrm{CAN}(k,t)$.

The infinite extensions of the first three rows and the first two columns of Table 1 are known precisely, as are the thirteen other exact values given in the table. Thus, the values of $\mathrm{CAN}(k,t)$ have been determined when either $t \geq k-2$ or $t \leq 2$. Other than these, that is, for $k \geq 3$ and $3 \leq t \leq k-3$, only thirteen values are known. They are: $\mathrm{CAN}(6,3) = \mathrm{CAN}(7,3) = \ldots = \mathrm{CAN}(11,3) = 12$, $\mathrm{CAN}(12,3) = 15$, and $\mathrm{CAN}(7,4) = \mathrm{CAN}(8,4) = \ldots = \mathrm{CAN}(12,4) = 24$. (See section 3.3.) For other values of t and k, we must settle for intervals delineated by lower and upper bounds for $\mathrm{CAN}(t,k)$.

3.1. Some basic results.

3.1.1. Some simple but useful inequalities are:
(a) $\mathrm{CAN}(k+1,t) \geq \mathrm{CAN}(k,t)$,
(b) $\mathrm{CAN}(k+1,t+1) \geq 2\,\mathrm{CAN}(k,t)$, and
(c) $\mathrm{CAN}(k,t) \geq 2^t$.

For (a), note that if $T \subseteq \{0,1\}^{k+1}$ is an $(s+1)$-face transversal in $[0,1]^{k+1}$, then the image $\pi_{[k]}(T)$ is an s-face transversal in $[0,1]^k$, so $\mathrm{CAN}(k+1,t) \geq \mathrm{CAN}(k,t)$. For (b) note that each facet of $[0,1]^{k+1}$ is itself a cube of dimension k, and any s-face transversal of $[0,1]^{k+1}$ must contain an s-face transversal of any two opposite facets, so $\mathrm{CAN}(k+1,t+1) \geq 2\,\mathrm{CAN}(k,t)$.

Also, (c) holds, since, for any t-surjective set S, the projection of S to a t-face must consist of all of the 2^t vertices of that face.

3.1.2. We have
(a) $\mathrm{CAN}(k,1) = 2$ for each $k \geq 1$,
(b) $\mathrm{CAN}(k,k) = 2^k$ for each $k \geq 1$, and
(c) $\mathrm{CAN}(k,k-1) = 2^{k-1}$ for each $k \geq 2$.

It is clear that each $(k-1)$-face (facet) of the cube contains either the zero vector or the vector of 1's, so (a) holds.

For (b), notice that if S is a k-surjective set, each vertex of the cube $[0,1]^k$ must be in S, so S must consist of all 2^k vertices of $[0,1]^k$.

Let S be the set of vertices $x = (x_1,\ldots,x_k) \in \{0,1\}^k$ of $[0,1]^k$ for which $\sum_i x_i$ is even. Then it is easy to see that S is a $(k-1)$-surjective set having 2^{k-1} elements, so that $\mathrm{CAN}(k,k-1) \leq 2^{k-1}$. The reverse inequality is 3.1.1(c) with $t = k-1$.

Suppose that A is an element of $\mathbf{CA}(n_1,k-1,t)$ and $B \in \mathbf{CA}(n_2,k-1,t-1)$. Then
$$\begin{pmatrix} A & 0 \\ B & 1 \end{pmatrix},$$
where the 0 represents the column vector of length n_1 having each entry 0 and the 1 represents the column vector of length n_2 having each entry 1, is an element of $\mathbf{CA}(n_1+n_2,k,t)$. This construction yields the following inequality. (See Theorem 2(i) of [41]. Theorem 3 of that paper presents a generalization.)

3.1.3. $\mathrm{CAN}(k+1,t) \leq \mathrm{CAN}(k,t) + \mathrm{CAN}(k,t-1)$.

The following inequality is a further strengthening of 3.1.3. It appears as the binary case of the first inequality of Theorem 3.2 of [30].

3.1.4. $\mathrm{CAN}(k+1,t) \leq \mathrm{CAN}(k,t) + 2\mathrm{CAN}(k-1,t-2)$.

3.2. The cases $t = 2$ and $s = 2$. The following statement was established in different ways in several papers independently, around 1970.

3.2.1. We have $\mathrm{CAN}(k,2) = n$, where n is the least positive integer such that $\binom{n-1}{\lceil \frac{n}{2} \rceil} \geq k$.

We describe one of the proofs. Letting $k = \binom{n-1}{\lfloor \frac{n}{2} \rfloor}$, it is possible to construct an example showing $\mathrm{CAN}(k,2) \leq n$, as follows. Construct an $n \times k$ matrix M. The first row consists of 0's; the remainder of M is the $(n-1) \times k$ matrix consisting of the $\binom{n-1}{\lceil \frac{n}{2} \rceil}$ possible columns having $\lceil \frac{n}{2} \rceil$ 1's and otherwise 0's.

When n is even, the argument showing that if $\mathbf{CA}(n,k,2) = \emptyset$ then $k \geq \binom{n-1}{\lfloor \frac{n}{2} \rfloor}$, already described in [68], is simple and involves Sperner's Lemma [77]; in [48], when n is odd, a more difficult argument shows that the Erdős-Ko-Rado Theorem [34] may be used, replacing Sperner's Lemma. We state these two results.

Sperner's Lemma. Let C be a collection of subsets of a set of cardinality m such that no element of C is a subset of another. Then the number of elements of C is at most $\binom{m}{\lfloor \frac{m}{2} \rfloor}$. The collection C consisting of all subsets having exactly $\lfloor \frac{m}{2} \rfloor$ elements achieves this bound.

Erdős-Ko-Rado Theorem. Let C be a family of subsets of $\{1,\ldots,a\}$ each having b elements, where $b \leq \frac{a}{2}$. Suppose that $A, B \in C$ implies $A \cap B = \emptyset$. Then C has at most $\binom{a-1}{b-1}$ elements. For a collection C achieving the bound, one may take all of the subsets $S \subseteq \{1,\ldots,a\}$ such that $|S| = b$ and $1 \in S$.

For proofs and generalizations of both of these theorems as well as of the LBYM Inequality mentioned later, see [42].

We give the argument first in the case of n even. Suppose T is a 2-independent family of k subsets of $\{1,\ldots,n\}$. Then no element of the family $T^* = T \cup T'$ consisting of the elements of T and their complements is a subset of a different element. Therefore, by Sperner's Lemma, $2k \leq \binom{n}{\frac{n}{2}}$. Then $k \leq \frac{1}{2}\binom{n}{\frac{n}{2}} = \binom{n-1}{\frac{n}{2}}$, which is $\binom{n-1}{\lceil \frac{n}{2} \rceil}$.

In case n is odd, letting T and T^* be as above, one has that the collection of elements of T^* having cardinality less than $\frac{n}{2}$ has cardinality k and has the property that no two elements have empty intersection. Katona [48] then shows that, in the extremal case, it may be assumed that the elements of this set are all of cardinality $\frac{n-1}{2}$. Therefore the Erdős-Ko-Rado Theorem applies and yields $k \leq \binom{n-1}{\frac{n-1}{2}-1}$, which equals $\binom{n-1}{\lceil \frac{n}{2} \rceil}$.

This result was discovered by several people independently, at about the same time. Rényi's book [68] contains this result, both for n even and n odd; for n even, the proof is given, and it is noted that Katona has a simple solution, using the Erdős-Ko-Rado Theorem, for n odd. (See problem P.1.8 of [68].) Katona's solution appears in [48]. The result was also obtained by Bollobas [10], Brace and Daykin [12], and Kleitman and Spencer [50]. Schönheim [71] proves a result that easily implies the result in case n is odd (the more difficult case): Given k subsets of a set with $2m$ elements, with no one containing another and with no two having empty intersection, the inequality $k \leq \binom{2m}{m-1}$ holds. This implies 3.2.1 when n is odd, since, if C is a 2-independent family of subsets of $\{1, \ldots, n\}$, then upon replacing each set $C \in C$ which has n as an element by its complement, one obtains a family of subsets of $\{1, \ldots, n-1\}$ such that none is contained in another and no two have nonempty intersection. Brace and Daykin [12] prove both of these results, and study many related statements.

3.2.2. We have $\mathrm{CAN}(k, k-2) = \lfloor 2^k/3 \rfloor$. The $(k-2)$-face transversal T having this cardinality is unique, up to symmetries of the cube.

As proven by Tang and Chen [79], the inequality $\mathrm{CAN}(k, k-s) \leq 2^k/(s+1)$ holds for all integers s and k such that $0 \leq s < k$. To see this, notice that $\mathrm{vert}([0,1]^{\mathrm{k}})$ is the union of the $s+1$ pairwise-disjoint sets $S_0, \ldots, S_s$, where S_j consists of those elements $x = (x_1, \ldots, x_k)$ of $\{0,1\}^k$ such that $x_1 + \cdots + x_k$ is congruent to j modulo $s+1$. Each set S_j is an s-face transversal. Clearly, at least one of the sets must have cardinality at most $2^k/(s+1)$. For a refinement of this bound see 4.4.2.

To establish that $\lfloor \frac{2^k}{3} \rfloor$ is a lower bound when $s = 2$ is more difficult. In this case, it is necessary to show that no 2-face transversal has fewer than $\lfloor \frac{2^k}{3} \rfloor$ vertices of the k-cube. Kostočka [51] and Johnson and Entringer [46], independently, each give a proof that $\mathrm{CAN}(k, k-2) = \lfloor \frac{2^k}{3} \rfloor$ by induction on k, simultaneously showing the uniqueness up to symmetry of the minimizing example.

3.3. The other thirteen values. The thirteen known values of CAN not covered already are dictated by the values already described, the inequalities of 3.1.1(a) and (b), an upper bound on $\mathrm{CAN}(12, 4)$ which is the result of the existence of an element of $\mathbf{CA}(24, 12, 4)$, and the result that $\mathrm{CAN}(12, 3) = 15$.

An array in $\mathbf{CA}(24, 12, 4)$ was first produced by an exhaustive search procedure described in Yan and Zhang [85]. It appears in Table 2. Subsequently, Colbourn and Kéri [29] discovered a simple construction for such an array.

An element of $\mathbf{CA}(15, 12, 3)$ was found by Nurmela [66] using the method of "tabu search"; so $\mathrm{CAN}(12, 3) \leq 15$. The inequality $\mathrm{CAN}(12, 3) \geq 15$ is due to Colbourn, Kéri, Rivas Soriano, and Schlage-Puchta [30], by exhaustive search. That paper also contains results on complete enumeration of the optimal covering arrays in several small cases.

4. Bounds on CAN.

In this section, we describe upper and lower bounds that have been obtained for the numbers $\mathrm{CAN}(k, t)$ (or equivalently, lower and upper bounds on $\mathrm{CAK}(n, t)$).

4.1. Lower bounds on CAN. When we move to the right in Table 1, the entry at least doubles, according to 3.1.1(b). Therefore we have the following inequality.

4.1.1. For $t \geq t_0$, $\mathrm{CAN}(k, t) \geq 2^{t_0}\mathrm{CAN}(k - t_0, t - t_0)$.

In Table 1, most of the lower bounds can be obtained by letting $t_0 = t - 2$ in 4.1.1.

This inequality indicates a connection between covering arrays and orthogonal arrays. When equality holds in 4.1.1, there is an element C of $\mathbf{CA}(n, k, t)$, where

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	1	1	1	1	1
3	0	0	1	0	0	1	1	0	0	1	1	1
4	0	0	1	1	1	0	1	0	1	0	0	1
5	0	1	0	0	1	0	1	1	0	0	1	1
6	0	1	0	1	0	1	1	0	1	0	1	0
7	0	1	1	0	0	1	0	1	1	0	0	1
8	0	1	1	1	0	0	1	1	0	1	0	0
9	1	0	0	0	1	1	0	0	1	0	1	1
10	1	0	0	1	0	1	1	1	0	0	0	1
11	1	0	1	0	0	0	1	1	1	0	1	0
12	1	0	1	1	0	1	0	0	1	1	0	0
13	1	1	0	0	0	0	1	0	1	1	0	1
14	1	1	0	1	1	0	0	1	1	0	0	0
15	1	1	1	1	0	0	0	0	0	0	1	1
16	1	1	1	0	1	1	1	0	0	0	0	0
17	0	0	0	0	1	1	1	1	1	1	0	0
18	0	0	1	1	1	1	0	1	0	0	1	0
19	0	1	0	1	1	1	0	0	0	1	0	1
20	0	1	1	0	1	0	0	0	1	1	1	0
21	1	0	0	1	1	0	1	0	0	1	1	0
22	1	0	1	0	1	0	0	1	0	1	0	1
23	1	1	0	0	0	1	0	1	0	1	1	0
24	1	1	1	1	1	1	1	1	1	1	1	1

Table 2. An Element of $\mathbf{CA}(24, 12, 4)$.

$n = 2^{t_0}\mathrm{CAN}(k - t_0, t - t_0)$. In each $n \times t_0$ submatrix of C, each t_0-tuple must occur exactly $\lambda = \mathrm{CAN}(k - t_0, t - t_0)$ times. That is, C is an $n \times k$ orthogonal array of strength t_0 having $\lambda = \mathrm{CAN}(k - t_0, t - t_0)$ and $n = 2^{t_0}\lambda$. This fact can be used in conjunction with a bound for orthogonal arrays due to Friedman [38] in the binary case (and generalized in Bierbrauer [6] for arbitrary v) to show that only the first two rows of Table 1 end in an infinite sequence having each entry twice the preceding entry.

Bierbrauer-Friedman Bound (in the binary case): Suppose there is a binary $n \times k$ orthogonal array of strength t. Then $n \geq 2^k(1 - \frac{k}{2(t+1)})$.

For example, this bound shows that there is no binary $6 \cdot 2^6 \times 11$ orthogonal array of strength 6 (for which λ would be 6), so it follows from the remark above that $\mathrm{CAN}(11, 8) > 6 \cdot 2^6$.

Kleitman and Spencer [50] derive two lower bounds on CAN. For the stronger of these, they make use of the LBYM Inequality.

The LBYM Inequality. Let H denote a family of subsets of S, no one containing another, where $|S| = m$. For $0 \le j \le m$, let h_j denote the number of sets in H having cardinality j. Then

$$\sum_{j=0}^{m} \frac{h_j}{\binom{m}{j}} \le 1.$$

The name of the inequality honors its four independent discoverers. See the original papers, Lubell [58], Bollobas [9], Yamamoto [84], and Meshalkin [62]; for the result and extensions, see section 4 of [42], where it is called the LYM Inequality. (At least, at the present time, only four discoverers are known to us! We obtained the reference to [9], and the name, from [49]. It is an often-noted curious fact that it is not unusual for mathematical results to have multiple discoverers, as is the case for the LBYM Inequality, as well as for 3.2.1 and 3.2.2.)

The stronger bound of Kleitman and Spencer is as follows.

4.1.2. If a binary covering array with parameters n, k, and t exists, where n is a multiple of 2^{t-1}, then

$$\genfrac{}{}{0pt}{}{k}{t-2} \le \genfrac{}{}{0pt}{}{n}{\frac{n}{2^{t-1}}+1} \Big/ (2^{t-3} \genfrac{}{}{0pt}{}{\frac{n}{2^{t-2}}}{\frac{n}{2^{t-1}}+1}).$$

To establish this bound, they argue as follows. Let F denote a t-independent family of k subsets of $\{1, \ldots, n\}$. Let F^* denote the collection consisting of the elements of F and their complements. Kleitman and Spencer show that if B_i $(1 \le i \le t-2)$ are elements of F^* and if $B_1 \cap \cdots \cap B_{t-2}$ is of cardinality σ, then at least half of the $(\lfloor \frac{\sigma}{2} \rfloor + 1)$-element subsets of this intersection are contained in no other element of F^*. Let H denote the collection of subsets H of $\{1, \ldots, n\}$ which are contained in precisely $t-2$ of the elements of F^*, say, $B_1, \ldots, B_{t-2}$, and such that $|H| = \lfloor \frac{\sigma}{2} \rfloor + 1$, where $\sigma = |B_1 \cap \ldots \cap B_{t-2}|$. Clearly no set in H contains another. Let h_i denote the number of elements of H having cardinality i. Let x_p denote the number of intersections $B_1 \cap \cdots \cap B_{t-2}$ having cardinality p. One has $h_{\lfloor p/2 \rfloor + 1} \ge \frac{1}{2} x_p \binom{p}{\lfloor p/2 \rfloor + 1}$. Therefore, by the LBYM Inequality,

$$1 \ge \sum \frac{1}{2} x_p \genfrac{}{}{0pt}{}{p}{\lfloor p/2 \rfloor + 1} \Big/ \genfrac{}{}{0pt}{}{n}{\lfloor p/2 \rfloor + 1} .$$

Also,

$$\sum x_p = \genfrac{}{}{0pt}{}{k}{t-2} 2^{t-2}$$

and

$$\frac{\sum p x_p}{\sum x_p} = \frac{n}{2^{t-2}}.$$

By linear programming the inequality of 4.1.2 follows.

The other (weaker) bound of Kleitman and Spencer, given that a binary covering array with parameters n, k, and t exists, is

$$\begin{matrix}k\\t-1\end{matrix} \leq \sum_{j=0}^{\lfloor \frac{n}{2^{t-1}} \rfloor} \begin{matrix}n\\j\end{matrix} .$$

A comparable bound given under the same circumstance in problem P1.8(c) of Rényi's book is

$$k \leq t-2+\frac{1}{2} \begin{matrix}\lfloor n/2^{t-2} \rfloor\\ \lfloor \frac{1}{2}\lfloor n/2^{t-2}\rfloor\rfloor\end{matrix} .$$

Each of these can be used to obtain the fact that, for t fixed, there is a constant $c_t > 0$ such that $\mathrm{CAN}(k,t) \geq c_t \log_2(k)$. The best (largest) such constant currently known is obtained from 4.1.2. See 4.3.3.

4.2. Upper bounds on CAN**.** We include a brief discussion of the upper bounds of Table 1.

Some of the earliest construction techniques for covering arrays are from Roux [70]. One such construction yields the following inequality, from which several of the upper bounds for $\mathrm{CAN}(k,3)$ are derived.

4.2.1. $\mathrm{CAN}(2k,3) \leq \mathrm{CAN}(k,3) + \mathrm{CAN}(k,2)$.

To see this, note that if A is a binary covering array in $\mathbf{CA}(n_1,k,2)$ and $B \in \mathbf{CA}(n_2,k,3)$ then

$$\begin{matrix} A & A \\ B & J-B \end{matrix}$$

(where J is a matrix of 1's) is an element of $\mathbf{CA}(n_1+n_2,k,3)$.

Sloane [75] has noted that an element of $\mathbf{CA}(16,14,3)$ can be obtained from a Hadamard matrix of order 16, by removing two columns. Therefore, $\mathrm{CAN}(14,3)$ and $\mathrm{CAN}(13,3)$ are bounded above by 16.

The upper bounds in the last two columns are consequences of work of Colbourn, Kéri, Rivas Soriano, and Schlage-Puchta [30] and Torres-Jimenez and Rodriguez-Tello [82]. Many of the relevant covering arrays were found by simulated annealing. These bounds rather spectacularly improve upon those that appeared in an earlier version of the table.

Upper bounds not in the range covered by the table result from examples produced by various methods, many of which are discussed in Section 5.

4.3. The order of magnitude of CAN **as a function of** k**.** We would certainly like to know the values of

$$c_t = \liminf\Big(\frac{\mathrm{CAN}(k,t)}{\log_2 k}\Big)$$

and

$$d_t = \limsup\Big(\frac{\mathrm{CAN}(k,t)}{\log_2 k}\Big).$$

Positive lower bounds on the numbers c_t and upper bounds on the d_t's were given by Kleitman and Spencer, in [50]. With these numbers we have, for each t, as k goes to infinity,

4.3.1. $$(c_t - o(1))\log_2(k) \le \mathrm{CAN}(k,t) \le (d_t + o(1))\log_2(k)$$

and

4.3.2. $$2^{(\frac{1}{d_t}-o(1))m} \le \mathrm{CAK}(m,t) \le 2^{(\frac{1}{c_t}+o(1))m} \ ;$$

indeed, c_t, d_t are the "best possible" numbers satisfying this, in that if c_t is replaced by a larger number or if d_t is replaced by a smaller number in the above, then the appropriate inequality is no longer valid.

Below, expressions for the bounds for the c_t's and d_t's sometimes involve the "entropy function" $H(\alpha)$. For α between 0 and 1, the *entropy function* $H(\alpha)$ is

$$\begin{aligned} H(\alpha) &= \lim_{m\to\infty} \frac{1}{m} \log_2 \begin{matrix} m \\ \lfloor \alpha m \rfloor \end{matrix} \\ &= -\big(\alpha \log_2(\alpha) + (1-\alpha)\log_2(1-\alpha)\big). \end{aligned}$$

We may write $\binom{m}{\lfloor \alpha m \rfloor} = 2^{m(H(\alpha)+o(1))}$. For the genesis of the entropy function in information theory and its use in combinatorics, see section 5 of Spencer [76].

For $t > 2$, as noted by Kleitman and Spencer [50], it is not known whether or not c_t and d_t are equal, so there results the following problem.

PROBLEM 2. Does $\lim_{k\to\infty} \frac{\mathrm{CAN}(k,t)}{\log_2 k}$ exist?

The following inequality follows from 4.1.2.

4.3.3. $c_t \ge \frac{t-2}{H(\frac{1}{2^{t-1}}) - \frac{1}{2^{t-2}}}$.

These lower bounds on the c_t's from Kleitman-Spencer [50] haven't been improved upon.

4.3.4. Suppose the entries of an $n \times k$ matrix of 0's and 1's are chosen independently with equal probabilities of 0 and 1. Then the probability that the matrix is not in **CA**(n, k, t) is at most $2^t\binom{k}{t}(1-\frac{1}{2^t})^n$.

For a given t columns and a given t-tuple involving those columns, the probability that the t-tuple fails to occur in those columns is $(1-\frac{1}{2^t})^n$. There are 2^t such t-tuples and $\binom{k}{t}$ sets of t columns. The inequality follows from this. This argument, or the equivalent counting argument, has been given many times, as in Section 5 of Hartman [44], Kleitman and Spencer [50], and, perhaps first, in Nečiporuk [65].

4.3.5. From 4.3.4 it follows that

$$\mathrm{CAN}(k,t) \leq \frac{t}{\log_2\left(\frac{2^t}{2^t-1}\right)} \log_2(k)$$

and therefore that

$$d_t \leq \frac{t}{\log_2\left(\frac{2^t}{2^t-1}\right)}.$$

Writing

$$\log_2(\frac{2^t}{2^t-1}) = \frac{1}{\log_e 2} \log_e \frac{1}{1-\frac{1}{2^t}}$$

and expanding as an infinite series

$$\frac{1}{\log_e 2}(\frac{1}{2^t} + \frac{1}{2}\frac{1}{2^{2t}} + \frac{1}{3}\frac{1}{2^{3t}} + \cdots),$$

we see that the upper bound given for d_t is less than $t2^t \log_e 2$, and this is a good approximation for the given upper bound when t is large.

By restricting the choice of the columns of the matrix in the above argument to have $\lfloor \frac{n}{2} \rfloor$ 1's, Roux [70] was able to improve the upper bound when $t = 3$.

4.3.6.
$$d_3 \leq \frac{4}{4 - 2H(\gamma) - (2-\gamma)H(\frac{1}{2-\gamma})},$$

where $\gamma = \frac{1}{2}(3 - \sqrt{5})$.

This bound was also obtained in Graham, Harary, Livingston, and Stout [41].

Godbole, Skipper and Sunley [40] improved upon the upper bound for d_t when $t > 3$. They used a simple argument involving the Lovász Local Lemma (Erdős and Lovász, [35]) enabling them to cut a factor of $\frac{1}{t}$ off of the previous bound.

4.3.7.
$$d_t \leq \frac{t-1}{\log_2\left(\frac{2^t}{2^t-1}\right)}.$$

We describe the argument of [40].

The Symmetric Version of the Lovász Local Lemma. Let $A_1, \ldots, A_m$ be events in a probability space such that $Pr[A_i] \leq p$ for each i. Suppose that each A_i is independent of all but d of the others. If the product $ep(d+1)$ is at most 1, then $Pr[\bigcup_i A_i] = 1$ (where here e is the base of the natural logarithm, $e \approx 2.71828$.)

Let A_j be the event that the j-th set of t columns fails to contain one of the 2^t t-tuples of 0's and 1's. The probability of this is at most $2^t(1-\frac{1}{2^t})^n$; let $p = 2^t(1-\frac{1}{2^t})^n$.

Also it is clear that any set of t of the columns is disjoint from $\binom{k-t}{t}$ other such sets, so we may take $d = \binom{k}{t} - \binom{k-t}{t} - 1$. Then it is not difficult to show that for $n = \frac{t-1}{\log_2\left(\frac{2^t}{2^t-1}\right)} \log_2(k)(1+o(1))$, one has $ep(d+1) \leq 1$, so that the result follows by the Lovász Local Lemma.

Table 3 contains decimal approximations for the best of the known lower bounds for the c_t's and the upper bounds for the d_t's, when $t \leq 6$, as given above in 4.3.3, 4.3.6, and 4.3.7. The values ($c_t = d_t = 1$) for the case $t = 2$ are determined by 3.2.1.

t	2	3	4	5	6
c_t	1	3.21	6.81	14.1	29
d_t	1	7.6	32.2	87.3	220

Table 3. Bounds for the c_t's and d_t's.

4.4. The order of magnitude of $\mathrm{CAN}(s+t,t)$ **as a function of** t**.** We have seen that $\mathrm{CAN}(k,t)$ is bounded by a constant multiple of $\log k$ when t is held constant. When $s = k - t$ is fixed, $\mathrm{CAN}(s+t,t)$ exhibits exponential growth, since $\mathrm{CAN}(s+t,t) \geq 2^t$, as in 3.1.1(c).

4.4.1. The sequence $\frac{\mathrm{CAN}(s,0)}{2^s}, \frac{\mathrm{CAN}(s+1,1)}{2^{s+1}}, \frac{\mathrm{CAN}(s+2,2)}{2^{s+2}}, \ldots$ is increasing and bounded above by $\frac{1}{s+1}$.

The monotonicity here follows from 3.1.1(b). The upper bound is the binary case of a result of Tang and Chen [79]. See the argument following 3.2.2. It would certainly be nice to know the limit.

Problem 3. What is

$$\lim_{t\to\infty} \frac{\mathrm{CAN}(s+t,t)}{2^{s+t}}?$$

We have no evidence that the answer is not $1/(s+1)$; see also Problem 4, below.

Alon, Krech, and Szabó [4] study the maximum number p of colors in a coloring of the vertices of the k-cube such that each s-face has at least one vertex of each color. Such a coloring is called an *s-polychromatic coloring.* They show that as k goes to infinity, p approaches $\frac{1}{s+1}$. They also show that for any $p < \frac{2^s}{2s}$ there is an s-polychromatic coloring of the k-cube with $k \approx \frac{1}{2}e^{2^s/2sp}$. Consequently, for any $\epsilon > 0$ and $k \leq \frac{1}{2}e^{2^{(1-\epsilon)s}}$, $\mathrm{CAN}(k, k-s) \leq \frac{2s}{2^{\epsilon s}}2^k$.

Define a function $B(k,s,m)$ as follows:

$$B(k,s,m) = \sum_{\substack{j \equiv m \bmod (s+1),\\ 0 \leq j \leq k}} \begin{matrix} k \\ j \end{matrix} .$$

Then $B(k, s, m)$ gives the number of vectors in $\{0, 1\}^k$ for which the sum of the entries is congruent to m modulo $s + 1$. For any m, the set of such vectors is an s-face transversal. Therefore, $\mathrm{CAN}(s + t, t) \leq B(s + t, s, m)$ for each $m = 0, 1, \ldots, s$. We denote the minimum by $B_{min}(k, s)$:

$$B_{min}(k, s) = \min_{m} B(k, s, m).$$

It can be shown that it is achieved when $m = \lfloor \frac{k-s-1}{2} \rfloor$. This minimum has been studied by Johnson, Grassl, McCanna, and Székely [47]. One has the following bound on values of CAN.

4.4.2. For any s, t,

$$\mathrm{CAN}(s + t, t) \leq B_{min}(s + t, s),$$

where, letting $\sigma = 0$ if t is even and $\sigma = 1$ if t is odd,

$$B_{min}(k, s) = \frac{2^k}{s+1} + \frac{2^{k+1}}{s+1} \sum_{j=1}^{\lfloor \frac{s}{2} \rfloor} (-1)^j \cos^{k+\sigma}\left(\frac{j\pi}{s+1}\right).$$

For example, for $s = 3$, we get

$$B_{min}(t + 3, 3) = \begin{cases} 2^{t+1}(1 - 2(\frac{\sqrt{2}}{2})^{3+t}) & \text{t odd,} \\ 2^{t+1}(1 - 2(\frac{\sqrt{2}}{2})^{4+t}) & \text{t even.} \end{cases}$$

These numbers may also be computed by the following recursive construction. Start with the vector of binomial coefficients: $X_0 = (\binom{s}{0}, \binom{s}{1}, \ldots, \binom{s}{s})$. Given $X_j = (x_0, \ldots, x_s)$, let $X_{j+1} = (x_0 + x_1, x_1 + x_2, \ldots, x_s + x_0)$. Then $B_{min}(k, s)$ is the minimum of the entries of X_{k-s}. For example, to compute $B_{min}(9, 3)$, we have

$$\begin{aligned} X_0 &= (1, 3, 3, 1), \\ X_1 &= (4, 6, 4, 2), \\ X_2 &= (10, 10, 6, 6), \\ X_3 &= (20, 16, 12, 16), \\ X_4 &= (36, 28, 28, 36), \\ X_5 &= (64, 56, 64, 72), \\ X_6 &= (120, 120, 136, 136); \end{aligned}$$

so $B_{min}(9, 3)$ is 120, the minimum entry in X_6.

Problem 4. Does there exist, for each fixed $s \geq 1$, an integer t_s such that for $t \geq t_s$, $\mathrm{CAN}(s + t, t) = B_{min}(s + t, s)$?

This question is open for $s \geq 3$.

5. Methods of Finding Covering Arrays.

In this section, we give a brief overview of computational methods for finding covering arrays. Our interest is in the methods used. These methods include both constructions and search techniques. For a recent survey of software making use of many of these methods, see Grindal, Offutt, Andler [43]. Note that many of the papers to which we refer concern more general situations than the case of binary covering arrays. Most methods we discuss have been used primarily on cases in which the strength t is small, for which relatively small covering arrays exist.

Given k and t, exhaustive search can be used to find $n \times k$ covering arrays of strength t, with n as small as possible, and thereby to compute $\mathrm{CAN}(k,t)$. The method of Yan and Zhang [85], which incorporated several techniques to speed up the search process, found the covering array of Table 2. Another exhaustive search method, presented in the recent paper of Bracho-Rios, Torres-Jimenez, and Rodriguez-Tello [13], uses branch-and-bound techniques and has yielded good results. However, exhaustive search techniques become impractical for finding $n \times k$ covering arrays, even when k is still quite small. Therefore methods which produce small but not necessarily smallest covering arrays become desirable. We report on several such methods in what follows. It is difficult to judge with any accuracy how close these methods come to actually producing covering arrays of smallest size. Indeed Nayeri, Colbourn, and Konjevod [64] report good results with a method which starts with small covering arrays determined by various methods and attempts to produce smaller covering arrays having the same number of columns by changing a few values in order to make possible the elimination of some of the rows.

5.1. Incremental construction methods. Various methods for building up an array a little at a time have been tried. Sherwood [73] used the following approach. The covering array is constructed one row at a time by choosing at each iteration a row which maximizes the number of previously uncovered t-tuples that are covered. Using 4.3.4, this technique can be shown to produce covering arrays for which the number of rows satisfies the bound of 4.3.5. The iterative step, finding a row that maximizes the number of newly covered t-tuples, appears to be computationally difficult; indeed, it is NP-hard; see Problem H in Section 6. Therefore, although this method is useful well beyond the range of usefulness of exhaustive search, it is practical only when k is fairly small.

Another method, described in Cohen, Dalal, Fredman, and Patton [25], also finds covering arrays one row at a time, but circumvents the difficulty of finding the minimizing row by using a heuristic greedy technique with randomness to choose the new row. Because it uses a heuristic to find the new row, it can't be guaranteed to produce arrays for which the number of rows is within the bound of 4.3.5. However it can produce covering arrays for somewhat larger values of k for which the number of rows is comparatively small.

Instead of adding one row at a time, Lei and Tai [56], [78] suggest beginning with a covering array with fewer columns and at each iteration performing two steps to achieve a covering array with an additional column. In the first step a new column is added. The column is chosen to maximize the number of new t-tuples covered. (In the $t = 2$

case originally considered, this is done by an effective polynomial-time procedure.) In the second step, rows are added as necessary to complete the coverage, obtaining a new covering array having one more column. The original algorithm was described for pairwise coverage ($t = 2$). The general case is considered in Lei, Kacker, Kuhn, Okun, Lawrence [55] and Forbes, Lawrence, Lei, Kacker, Kuhn [36]. As was the case for the previous method, there is no guarantee that the covering array produced will have a number of rows within the bound of 4.3.5, but it can produce covering arrays for k fairly large in which the number of rows is comparatively small.

5.2. Heuristic search techniques. Various heuristic search techniques have been tried for the problem of finding small covering arrays.

Nurmela [66] uses tabu search to attempt to find covering arrays in $\mathbf{CA}(n, k, t)$. Tabu search is a heuristic search technique in which a tabu list is used to avoid cycling and broaden the searched region. In Nurmela's setup, the search space is the set of $n \times k$ matrices of 0's and 1's and the cost function is the number of uncovered t-tuples. A move consists of changing precisely one entry in such a way as to cover a specified uncovered t-tuple. (In the initial stage this may not be possible, in which case several changes are made.) The tabu list contains the last T entries that have been changed, so that changing any of these is disallowed. The move is chosen randomly from those that achieve the minimum cost that are not on the tabu list. Values of T in the range $1, \ldots, 10$ were found to be useful. Nurmela's covering arrays can be found at the website, [67]. For another use of tabu search for covering arrays, see Walker and Colbourn [83].

Simulated annealing is a heuristic search technique modeled in analogy to the annealing process for solids as studied in statistical mechanics. It involves a cost function defined on the search space of feasible solutions, a method of producing neighbors of a given feasible solution, and a current temperature τ, which is reduced as the method progresses, until an equilibrium situation results. At each stage, a neighboring solution to the current solution is produced at random. It immediately replaces the current solution provided that its cost is lower. If its cost is higher, the following process is used to decide whether or not the neighboring solution will replace the current solution. A number δ is chosen at random uniformly from the interval $[0, 1]$. If $\delta \leq e^{-\Delta/\tau}$, where Δ is the increase in the value of the cost function, then the neighboring solution becomes the current solution. This is known as Metropolis's criterion. The function $e^{-\Delta/\tau}$ is analogous to Boltzmann's probability of statistical mechanics.

Simulated annealing for covering arrays has been studied in Cohen, Colbourn, Gibbons, and Mugridge [23], where the cost function was the number of uncovered t-tuples and the temperature τ was decreased by a constant factor near 1 at each iteration. In Cohen, Colbourn, and Ling [24], this approach was used in combination with a construction technique. More recently, Torres-Jimenez and Rodriguez-Tello [82] have used simulated annealing, successfully determining a number of binary covering arrays of strengths $t = 3$ to 5. In [69], the same authors studied a memetic algorithm built upon their work with simulated annealing and enlisted this algorithm in the search for covering arrays. Memetic algorithms perform heuristic searches using rules based loosely upon evolutionary theory.

Several other heuristic search techniques, including hill climbing, ant crawl, great flood, and genetic algorithms have been tried. See [43] for references.

Incidentally we note that, motivated by the difficulties involved in verifying that large arrays are covering arrays, Avila-George, Torres-Jimenez, Hernandez, and Rangel-Valdez [2] introduce an algorithm utilizing grid computing to do this.

5.3. Methods using codes and similar constructions. The first constructive method which, for fixed t, produced $n \times k$ covering arrays of strength t with n bounded by a constant multiple of $\log k$ was obtained by Alon in [1] utilizing error-correcting codes. This solved a problem of Kleitman and Spencer [50]. Although the constant was sufficiently large to make his arrays impractical for applications, Alon's method is the prototype for the use of perfect hash functions. (See below.)

Sloane [75], also using codes, concentrated on the binary, strength 3 case. An (m, d, k) *binary code* is a matrix of size $m \times k$ having entries from $\{0, 1\}$ such that each pair of rows disagree in at least d entries. The code is called *intersecting* if each pair of rows have the common value 1 in at least one entry. Intersecting codes are closely related to covering arrays of strength three. Sloane showed (Theorem 3 of [75]) that, given a binary intersecting code, one obtains a covering array of strength 3 by adding a row consisting of 0's and a row consisting of 1's. The paper [75] also announces a polynomial-complexity algorithm for constructing binary $n \times k$ covering arrays of strength 3 having n bounded by about $12.3 \log_2(k)$. This is asymptotically a smaller value of n than is currently achieved by other known methods; the coefficient 12.3 is larger than Roux's bound $d_3 \leq 7.6$ of Table 3, but smaller than the Kleitman-Spencer bound $d_3 \leq 15.6$ of 4.3.5.

Roux's result 4.2.1 by itself is a practical method for constructing arrays of strength 3 which yields small arrays when k is not large, as we have noted above in Section 4.2. In general this method yields covering arrays for which n exceeds that produced by Sloane's method, being asymptotic to $\frac{1}{2}(\log_2(k))^2$. Recall that, given an $n_1 \times k$ covering array of strength 3 and an $n_2 \times k$ covering array of strength 2, Roux's construction yields an $n \times 2k$ covering array of strength 3, where $n = n_1 + n_2$. We know that $\mathrm{CAN}(k, 2) \approx \log_2(k)$, so if $f(k)$ denotes the size of the best covering array produced by recursive use of this method, then $f(2k) \approx f(k) + \log_2(k)$. From this it follows that, with $k = 2^\ell$, $f(k) \approx (\ell - 1) + (\ell - 2) + \cdots + 1 = \binom{\ell}{2}$, so that $f(k) \approx \frac{1}{2}(\log_2 k)^2$ for large k.

A matrix of size $r \times s$ having entries from a set of m symbols is called *perfectly t-hashing*, where $t \leq s$, if for every set of t columns there is a row in which the t entries are distinct. Viewing each row as determining a function from the set $\{1, \ldots, s\}$ to the set of m symbols, the condition is that for each subset T of $\{1, \ldots, s\}$ of cardinality t, the restriction of one of the functions to T is injective. If A is an $n_1 \times N$ perfectly t-hashing array with entries from an m-element set S and B is an $n_2 \times m$ binary covering array of strength t whose columns correspond to the elements of S then the $n_1 n_2 \times N$ array C obtained from A by replacing each symbol by the corresponding column of B is a binary covering array of strength t. Perfect hash families derived from error-correcting codes have been used in this way by many authors. See, for example, Cohen and Zémor [26].

In Matirosyan and Trung [60], use of perfect hash families yields an improvement over a bound of Godbole, Skipper, and Sunley [40] of which 4.3.7 is the (binary) case $v = 2$, when $v > 2$. Recently, matrices satisfying somewhat weaker conditions than those required by the property of perfectly t-hashing have been found useful in constructing covering arrays. Colbourn and Torres-Jimenez [32] have considered generalizations in which rows of the matrix can have entries from sets of different sizes, and have derived covering arrays (including many in the binary case) that improve upon the sizes of previously-known arrays.

Much work was motivated by Naor and Naor [63] which in Proposition 8.1 noted that k-wise δ-independent probability spaces yield binary covering arrays strength t, when δ is at most 2^{-t}. Alon, Bruck, Naor, Naor, and Roth [3] applied expander graphs to this problem. See also Bierbrauer and Schellwat [8].

The "squaring construction," theorem 7.4 of Hartman [44], is based upon a construction involving Turán's theorem of graph theory. Given a binary covering array of strength t having k columns and n rows (an element of $\mathbf{CA}(n, k, t)$), it yields an element of $\mathbf{CA}(qn, k^2, t)$, where $q = \lfloor \frac{t^2}{4} \rfloor + 1$.

Colbourn, Martirosyan, Trung, and Walker [31] give many recursive constructions. See the survey [27], where Colbourn describes many more construction methods.

Sherwood, Martirosyan, and Colbourn [74] describe a condition that determines whether or not a matrix consisting of permutations of the vectors in a finite field is a covering array. They make use of this condition and perfect hash families to produce covering arrays of strengths $t = 3, 4$. Walker and Colbourn [74] use permutation vectors together with tabu search to find covering arrays.

Colbourn and Kéri [29] study the relationship between binary covering arrays and existentially closed graphs. A *t-existentially closed graph* is a graph such that, whenever sets A, B form a partition of a set of t vertices of the graph, there is a vertex v not in $A \cup B$ such that each vertex in A is adjacent to v, while no vertex of B is adjacent to v. The adjacency matrix of a t-existentially closed graph with k vertices is an element of $\mathbf{CA}(k, k, t)$. Several constructions of t-existentially closed graphs are noted. They also introduce a bipartite analogue of this notion which relates in a similar way to elements of $\mathbf{CA}(n, k, t)$, where n need not equal k.

5.4. Probabilistic methods and derandomization. For any fixed p $(0 < p \leq 1)$, letting n be an integer larger than $(t(1 + \log_2(k)) + \log_2(1/p))/ \log_2(2^t/(2^t - 1))$, choose the entries of an $n \times k$ array C by coin flipping. Then, using 4.3.4, the probability that C is a covering array of strength t is more than $1 - p$.

This idea can be used, one-row-at-a-time. Let D be an arbitrary set of t-tuples, which perhaps could be the set of t-tuples which remain uncovered thus far, and let x be an element of $\{0, 1\}^k$, chosen "at random, by coin-flipping." Each t-tuple of D is covered by exactly 2^{k-t} such tests x, so the probability that such a t-tuple is covered by x is 2^{-t}. Therefore, the expected number of t-tuples covered by x is $|D|/2^t$. This implies that there certainly exist tests x which cover at least $|D|/2^t$ elements of D. If a reasonable method of choosing such a test x is devised, then, starting with D consisting of all the t-tuples, of which there are $\binom{k}{t}2^t$, the cardinality of the uncovered t-tuples is multiplied by a factor of $1 - \frac{1}{2^t}$ at each step, and is reduced to less than one (and therefore to zero) in no more than $t(\log_2(k) + 1)/\log_2((2^t + 1)/2^t)$ steps.

The parallelizable method described in Kuhn [52] relies on this probability bound; it chooses a number of possible choices for the next test, from among which the best candidate is chosen. The possible choices are selected by use of a random number generator augmented by a method of "modular incrementing" to get additional tests at little computational cost.

Bryce and Colbourn [14] have devised a deterministic algorithm that produces covering arrays which achieve the Kleitman-Spencer bound 4.3.5 on $\mathrm{CAN}(k, t)$. This method is of polynomial complexity, for t fixed. It is described in the v-valued, strength $t = 2$ case in [14], and in [15] for higher strengths. Suppose that we are producing the covering array as above, so that D is the set of thus-far uncovered t-tuples. We wish to choose a new test that covers many t-tuples from D. We would certainly like to do as well as one would "expect" to do by picking the entries at random; and it is indeed possible to do this deterministically. Consider the experiment of choosing a random test T. Let X denote the random variable that gives the number of tuples of D left uncovered by T. Given a partly-filled-in test T', let $E(X|T')$ denote the conditional expectation of X given that the chosen test T extends T'. If T' is the test having no entries, then this conditional expectation is the number of the Kleitman–Spencer bound; if T' is a complete test, then there is no randomness left, and the number is simply the number of tuples of D which remain uncovered. Given any partially-filled-in test T' and entry $T'(i)$ as yet undetermined, we consider extending the test. We can extend the test either by choosing 0 for the new value (call the result T_0) or by choosing 1 (call this T_1). Then $E(X|T') = \frac{1}{2}E(X|T_0) + \frac{1}{2}E(X|T_1)$, so one of the two latter expectations is at most as large as $E(X|T')$. Therefore it is possible to choose a value for $T'(i)$ that does not increase the expected value. Letting T'' denote the extension, T_0 or T_1, that achieves this, we have $E(X|T'') \leq E(X|T')$.

It follows that if we start with the partial test with no entries, we can fill in the entries one-at-a-time, in any order, while never increasing this expected value; and in the end, we have a covering array satisfying the Kleitman-Spencer bound. Since it is not difficult to compute this expected value, this is a feasible approach. In actual practice, the expected values decrease, so that the result is usually a covering array of strictly smaller size than the smallest guaranteed by 4.3.5.

It would be nice to have a similar method which could provably achieve the stronger bound 4.3.7 of Godbole, Skipper, and Sunley [40]. This might involve the determination of a method to "derandomize" the use of the Lovász Local Lemma.

Problem 5. Find a deterministic algorithm which provably finds covering arrays for which the number of rows satisfies the Godbole, Skipper, Sunley bound and which runs with polynomial complexity.

6. Computational Complexity.

In this section we describe several decision problems related to the production of "small" covering arrays. The computational complexity for each problem is briefly discussed. We do not attempt to survey the computational complexity of the many published methods for finding covering arrays. The main objective is to correct a prevalent misconception concerning complexity and covering arrays.

Many articles contain statements of the form "Determining CAN(k, t) is NP-complete" and "Determining the smallest n for which there exists a v-valued, pairwise (strength $t = 2$) covering array with k columns and n rows is NP-complete," either without reference or referring to [72] or [56]. In fact, Seroussi and Bshouty [72] prove the NP-completeness of a more general decision problem - see Problem G, below. Also, in [56] it is stated that "the problem of generating a minimum test set for pairwise testing is NP-complete"; however the proof of this is erroneous, since the "pair-cover problem" as described in that paper fails to match up correctly with the problem of finding strength $t = 2$ covering arrays. These problems of determining NP-completeness seem to be open.

Problem A.

INPUT: Positive integers n, k, t with $t \leq k$. (The input requires $O(\log_2(nkt))$ space.)

OUTPUT: "Yes" if $\mathbf{CA}(n, k, t) = \emptyset$; "No" otherwise.

It is not known whether or not an algorithm of polynomial complexity exists for the problem. It is not known whether or not the problem is in NP. Exhibiting an element of $\mathbf{CA}(n, k, t)$ requires exponential time/space; the number of entries in such a matrix is not bounded by a polynomial in the input size, $\log_2(nkt)$. For this reason, the next problem may be of more interest.

Problem B. (The "unary" version of the preceding.)

INPUT: The numbers n, k, t in unary notation; equivalently for our purposes, an $n \times k$ matrix of 0's.

OUTPUT: "Yes" if $\mathbf{CA}(n, k, t) = \emptyset$; "No" otherwise.

This problem is in the class NP: If $\mathbf{CA}(n, k, t) = \emptyset$ then an element C in this set confirms this. It is not known whether or not there is an algorithm of polynomial complexity, or whether or not this problem is NP-complete.

The situation is the same for the versions of the above in which $t > 2$ is fixed. In particular, for $t = 3$, we have the following two problems.

Problem C.

INPUT: Positive integers n, k.

OUTPUT: "Yes" if $\mathbf{CA}(n, k, 3) = \emptyset$; "No" otherwise.

(For complexity comments, see Problem A.)

Problem D.

INPUT: Positive integers n, k, given in unary notation.

OUTPUT: “Yes” if $\mathbf{CA}(n, k, 3) = \emptyset$; “No” otherwise.

(For complexity comments, see Problem B.)

In the non-unary case, one might ask if there is a good algorithm to determine whether or not $\mathrm{CAN}(k, t)$ lies in a given range. In this regard we include the following two problems.

Problem E.

INPUT: Positive integers k, t with $t \leq k$.

OUTPUT: “Yes” if $\mathbf{CA}(n, k, t) = \emptyset$, where $n = \left\lfloor \frac{t \log_2 k}{\log_2\left(\frac{2^t}{2^t-1}\right)} \right\rfloor$; “No” otherwise.

The algorithm is easy: The output is always “Yes,” according to 4.3.5. The algorithm of Bryce and Colbourn [14] described in section 5.4 yields such a covering array in time polynomial in $k + t$. (Of course, no algorithm could produce, in the sense of filling in the entries, such a covering array in time polynomial in $\log_2(kt)$, since the number of entries in such an array is not bounded by a polynomial in $\log_2(kt)$.)

Problem F.

INPUT: Positive integers k, t with $t \leq k$.

OUTPUT: “Yes” if $\mathbf{CA}(n, k, t) = \emptyset$, where $n = \left\lfloor \frac{(t-1) \log_2 k}{\log_2\left(\frac{2^t}{2^t-1}\right)} \right\rfloor$; “No” otherwise.

This time the output should be “Yes,” when k is large enough (for fixed t), according to 4.3.7; however, no algorithm is known which produces a covering array of the given size in time polynomial in $k + t$.

The following problem is considered in Seroussi and Bshouty [72].

Problem G. (Problem $TS(n, t)$ of [72].)

INPUT: A positive integer k and a list $(R_1, \ldots, R_w)$ of subsets of $\{1, \ldots, k\}$, each having cardinality t. (The integers n and t are fixed, with $t \geq 1$, $n \geq 2^t$.

OUTPUT: “Yes” if there exists a set $S \subseteq \{0, 1\}^k$ of cardinality at most n such that the projection $\pi_{R_j}(S)$ of S to the coordinates represented by R_j equals $\{0, 1\}^t$ (surjective), for each j.

The problem $TS(2^t, t)$ is NP-complete, for each $t \geq 2$: It is shown in [72] that the problem is in the class NP; that the 3-coloring problem of graph theory is equivalent to $TS(4, 2)$; and that the problem $TS(4, 2)$ reduces to $TS(2^t, t)$, for each $t > 2$.

For recent progress on a further generalized version of Problem G, see Cheng [22].

Finally, the following concerns a sub-problem which arises in algorithms for constructing covering arrays one-row-at-a-time.

Problem H.

INPUT: Positive integers m, t and an $n \times k$ matrix A with entries in $\{0, 1\}$.

OUTPUT: “Yes,” if it is possible to extend A by one row and cover at least m t-tuples not originally covered.

This problem is NP-complete. See Colbourn, [27].

7. Notes and Acknowledgments.

We have restricted our attention to the binary case. Much research has been done on generalizations involving v-valued arrays, and on $(v_1, \ldots, v_k)$-valued arrays. Such work has largely been omitted from the present report.

Other more general questions are possible as well, even in the binary case. We mention a couple of these.

If, instead of requiring that in each set of t columns of the array all of the possible 2^t vectors occur (at least once), we insist that they occur at least λ times, we might ask for the minimum number of rows of such an array. This question was considered by Frankl in [37]. See Katona [49] for recent results, and Bulutoglu and Margot [17] for an even more recent use of integer programming techniques on such problems.

Meagher and Stevens [61] have considered the following question. A graph G having as its vertices the columns of the array is given, and it is required that for each pair of adjacent (in the graph) columns, all of the four possible 0-1 patterns appear. Given G, what is the smallest possible number of rows in such an array? Clearly it is not more than $\mathrm{CAN}(k, 2)$, and it is shown in [61] that strict inequality is possible.

Colbourn, Kéri, Rivas Soriano, and Schlage-Puchta [30] have introduced the notion of a *radius covering array.* The definition (in the binary case) involves, in addition to n, k, and t, a parameter $r < t$, and for each t-tuple, it is required that there exist a row of the array that agrees with that t-tuple in all but r entries. For $r = 0$ this reduces to the standard definition of a covering array. They give several methods to construct such arrays and describe results of their use.

We greatly appreciated comments on a prior version of this paper by Charles Colbourn and Renée Bryce. The current version also benefited from several very helpful reports by referees, as well as from a conversation with Noga Alon about covering arrays.

References.

[1] N. Alon, Explicit construction of exponential-sized families of k-independent sets. *Discret. Math.* **58** (1986), 191–193.

[2] H. Avila-George, J. Torres-Jimenez, V. Hernandez, and N. Rangel-Valdez, Verification of general and cyclic covering arrays using grid computing. *Data Management in Grid and Peer-to-Peer Systems, Third International Conference, Globe 2010.*

[3] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth, Construction of asymptotically good, low-rate error-correcting codes through random graphs. *IEEE Trans. on Inf. Theory 38* (1992), 509–516.

[4] N. Alon, A. Krech, and T. Szabó, Turàn's theorem in the hypercube. *SIAM J. Discret. Math.* **21** (2007), 66–72.

[5] B. Becker and H. U. Simon, How robust is the n–cube? *Inform. and Comput.* **77** (1988), 162–178.

[6] J. Bierbrauer, Bounds on orthogonal arrays and resilient functions. *J. Comb. Des.* **3** (1995), 179–183.

[7] J. Bierbrauer and H. Schellwat, Almost independent and weakly biased arrays: efficient constructions and cryptologic applications. *Adv. in Cryptol., CRYPTO 2000, Lect. Notes in Comput. Sci.* (2000), 533–543.

[8] J. Bierbrauer and H. Schellwat, Weakly biased arrays, almost independent arrays, and error-correcting codes. *Codes and Association Schemes (Piscataway, NJ, 1999)*, 33–46, DIMACS Ser. Discret. Math. Theor. Comput. Sci., 56, Amer. Math. Soc., Providence, RI, 2001.

[9] B. Bollobas, On generalized graphs. *Acta Math. Hung.* **16** (1965), 447–452.

[10] B. Bollobas, Sperner systems consisting of pairs of complementary subsets. *J. Comb. Theory A* **15** (1973), 363–366.

[11] S. Y. Boroday and I. S. Grunskii, Recursive generation of locally complete tests. *Cybern. and Syst. Anal.* **28** (1992), 20–25.

[12] A. Brace and D. E. Daykin, Sperner type theorems for finite sets. In *Combinatorics (Proc. Conf. on Comb. Math.)*, ed. D. J. A. Welsh and D. R. Woodall, Oxford, 1972, 18–37.

[13] J. Bracho-Rios, J. Torres-Jimenez, and E. Rodriguez-Tello, A new backtracking algorithm for constructing binary covering arrays of variable strength. *MICAI 2009: Advances in Artificial Intelligence, Lect. Notes in Comput. Sci.* **5845** (2009), 397–407.

[14] R. C. Bryce and C. J. Colbourn, The density algorithm for pairwise interaction testing. *J. Softw. Test., Verification, and Reliab.* **17** (2007), 159–182.

[15] R. C. Bryce and C. J. Colbourn, A density-based greedy algorithm for higher strength covering arrays. *J. Softw. Test., Verification, and Reliab.* **19** (2009), 37–53.

[16] R. C. Bryce, A. Rajan, and M. P. E. Heimdahl, Interaction testing in model-based development: effect on model coverage. *Proc. of Thirteenth Asia-Pacific Softw. Eng. Conf., Bangalore, India (APSEC '06)* (2006), 258–269.

[17] D. A. Bulutoglu and F. Margot, Classification of orthogonal arrays by integer programming. *J. Stat. Plan. and Inference* **138** (2008), 654–666.

[18] K. Burr and W. Young, Combinatorial test techniques: table-based automation, test generation, and code coverage. *Proc. of the Int. Conf. on Softw. Test. Anal. and Rev.*, October 1998, 503–513.

[19] P. Busschbach, Constructive methods to solve the problems of s-surjectivity, conflict resolution, coding in defective memories. *Report 84D005, École Nationale Supér. Télécomm.*, Paris, 1984.

[20] A. K. Chandra, L. T. Kou, G. Markowsky, and S. Zaks, On sets of Boolean n–vectors with all k-projections surjective. *Acta Inform.* **20** (1983), 103–111.

[21] M. A. Chateauneuf and D. L. Kreher, On the state of strength-three covering arrays. *J. Comb. Des.* **10** (2002), 217-238.

[22] C Cheng, The test suite generation problem: optimal instances and their implications. *Discret. Appl. Math.* **155** (2007), 1943–1957.

[23] M. B. Cohen, C. J. Colbourn, P. B. Gibbons, and W. B. Mugridge, Constructing test suites for interaction testing. In *Proc. of Int. Conf. on Softw. Eng. (ICSE 2003)*, Portland, Oregon, May, 2003, 38–49.

[24] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling, Constructing strength three covering arrays with augmented annealing. *Discret. Math.* **308** (2008), 2709–2722.

[25] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, The AETG system: an approach to testing based on combinatorial design. *IEEE Trans. on Softw. Eng.* **24** (1997), 437–444.

[26] G. D. Cohen and G. Zémor, Intersecting codes and independent families. *IEEE Trans. on Inf. Theory* **40** (1994), 1769–1780.

[27] C. J. Colbourn, Combinatorial aspects of covering arrays. *Le Matematiche (Catania)* **59** no. 1-2 (2004), 125–172.

[28] C. Colbourn, CA tables for $t = 2, 3, 4, 5, 6$.
`http://www.public.asu.edu/ccolbou/src/tabby/catable.html` .

[29] C. J. Colbourn and G. Kéri, Binary covering arrays and existentially closed graphs. *Lect. Notes in Comput. Sci.* **5557** (2009), Coding and Cryptography, 22–33.

[30] C. J. Colbourn, G. Kéri, P. P. Rivas Soriano, and J.-C. Schlage-Puchta, Covering and radius-covering arrays: construction and classification. *Discrete Applied Mathematics* **158** (2010), 1158–1180.

[31] C. J. Colbourn, S. S. Martirosyan, T. V. Trung, and R. A. Walker, II, Roux-type constructions for covering arrays of strengths three and four. *Des., Codes and Cryptogr.* **41** (2006), 33–57.

[32] C. Colbourn and J. Torres-Jimenez, Heterogeneous hash families and covering arrays. *Contemporary Mathematics* **523** (2010), 3–15.

[33] S. Dunietz, W. K. Ehrlich, B. D. Szablak, C. L. Mallows, and A. Iannino. Applying design of experiments to software testing. *Proc. Int. Conf. on Soft. Eng. (ICSE '97)* October 1997, 205–215.

[34] P. Erdős, C. Ko, and R. Rado, Intersection theorems for systems of finite sets. *Q. J. Math. Oxf. Ser.* (2), **12** (1961), 313–318.

[35] P. Erdős and L. Lovász, Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and Finite Sets*, ed. A. Hajnal, et al., Colloq. Math. Soc. J. Bolyai 11, North Holland, 1975.

[36] M. Forbes, J. Lawrence, Y. Lei, R. Kacker, and R. Kuhn, Refining the In-Parameter-Order strategy for constructing covering arrays. *NIST J. Res.* **113** (2008), no 5, 287–297.

[37] P. Frankl, An extremal problem of coding type. *Ars Comb.* **1**, (1976) 53–55.

[38] J. Friedman, On the bit extraction problem. In *33rd IEEE Symposium on the Foundations of Computer Science*, 1992, 314–319.

[39] S. Gal, Rendezvous search on a line. *Oper. Res.* **47** (1999), 974–976.

[40] A. P. Godbole, D. E. Skipper, and R. A. Sunley, t-covering arrays: upper bounds and Poisson approximations. *Comb., Probab., and Comput.* **5** (1996), 105–117.

[41] N. Graham, F. Harary, M. Livingston, Q. F. Stout, Subcube fault-tolerance in hypercubes. *Inf. and Comput.* **102** (1993), no. 2, 280–314.

[42] C. Greene and D. J. Kleitman, Proof techniques in the theory of finite sets. In *MAA Stud. in Math.*, vol. **17**, *Stud. in Comb.*, ed. G.-C. Rota, Math. Assoc. of America, 1978, pp. 22–79.

[43] M. Grindal, J. Offutt, and S. F. Andler, Combination testing strategies: a survey. *J. Softw. Test., Verification, and Reliab.* **15** (2005), 167–199.

[44] A. Hartman, Software and hardware testing using combinatorial covering suites. In *Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications*, eds. M. C. Golumbic and I. B. Hartman, Springer, 2005, pp. 237–266.

[45] A. S. Hedayat, N. J. A. Sloane, J. Stufken, *Orthogonal Arrays, Theory and Applications.* Springer, 1999.

[46] K. A. Johnson and R. Entringer, Largest induced subgraphs of the n-cube that contain no 4-cycle, *J. Comb. Theory, B* **46** (1989), 346–355.

[47] K. A. Johnson, R. Grassl, J. McCanna, and L. A. Székely, Pascalian rectangles module n, *Quaest. Math.* **14** (1991), 383–400.

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

[49] G. O. H. Katona, Strong qualitative independence. *Discret. Appl. Math.* **137**(2004), 87–95.

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

[51] E. A. Kostočka, Piercing the edges of the n-dimensional unit cube. (Russian.) *Diskret. Analyz.* Vyp. 28 Metody Diskretnogo Analiza v Teorii Grafov i Logiceskih Funcii (1976), 55–64, 79.

[52] D. R. Kuhn, An algorithm for generating very large covering arrays. NISTIR 7308 (2006); available at:
`http://nvl.nist.gov/pub/nistpubs/ir/2006/ir7308.pdf`.

[53] D. R. Kuhn and M. Reilly, An investigation of the applicability of design of experiments to software testing. *Proc. 27th Annu. NASA Goddard/IEEE Softw. Eng. Workshop*, October 2002, 91–95.

[54] D. R. Kuhn, D. Wallace, and A. Gallo, Software fault interactions and implications for software testing. *14-th IEEE Int. Conf. and Workshop on the Eng. of Computer-based Syst.*, 2007.

[55] Y. Lei, R. Kacker, R. Kuhn, V. Okun, and J. Lawrence, IPOG: a general strategy for t-way software testing. *Proc. of the 14-th Annu. IEEE Int. Conf. on the Eng. of Computer-based Syst.*, 2007.

[56] Y. Lei and K. C. Tai, In–parameter order: A test generation strategy for pairwise testing. In *Proc. Third IEEE High Assurance Syst. Eng. Symp.*, 1998, 254–261.

[57] W. S. Lim and S. Alpern, Minimax rendezvous on the line. *SIAM J. Control and Optim.* **34** (1996), 1650–1665.

[58] D. Lubell, A short proof of Sperner's theorem. *J. Comb. Theory* **1** (1966), 299.

[59] E. Marczewski, Indépendance d'ensembles et prolongement de mesures. *Colloq. Math.* **1** (1948), 122–132.

[60] S. Martirosyan and T. van Trung, On t-covering arrays. *Des., Codes and Cryptogr.* **32** (2004), 323–339.

[61] K. Meagher and B. Stevens, Covering arrays on graphs. *J. Comb. Theory B* **95** (2005), 131–151.

[62] L. D. Meshalkin, A generalization of Sperner's theorem on the number of subsets of a finite set. *Theor. Probab. Appl.* **8** (1963), 203-204.

[63] J. Naor and M. Naor, Small bias probability spaces: efficient constructions and applications. *SIAM J. Comput.* **22** (1993), 838–856.

[64] P. Nayeri, C. J. Colbourn, and G. Konjevod, Randomized postoptimization of covering arrays. *Proc. Int. Workshop on Comb. Algorithms (IWOCA2009), Lect. Notes in Comput. Sci.* **5874** (2009), 408–419.

[65] É. I. Nečiporuk, Complexity of gating circuits which are realized by Boolean matrices with undetermined elements (Russian). *Dokl. Akad. Nauk SSSR*, **163** (1965), 44–42. Translated in *Sov. Phys. - Dokl., Cybern. and Control Theory* **10** (1966), 591–593.

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

[67] K. Nurmela, Upper bounds for covering arrays by tabu search (webpage): `http://www.tcs.hut.fi/∼kjnu/covarr.html`.

[68] A. Rényi, *Foundations of Probability*, Holden-Day, 1970.

[69] E. Rodriguez-Tello and J. Torres-Jimenez, Memetic algorithms for constructing binary covering arrays of strength three. *LNCS 5975, pp. 86-97, 2010*

[70] G. Roux, *k–proprietes dans les tableaux de n colonnes: cas particulier de la k–surjectivite et de la k–permutivite*, Ph.D. Thesis, Universite de Paris, 1987.

[71] J. Schönheim, On a problem of Purdy related to Sperner systems. *Can. Math. Bull.* **17** (1974), 135–136.

[72] G. Seroussi, N. H. Bshouty, Vector sets for exhaustive testing of logic circuits. *IEEE Trans. on Inf. Theory*, **34** (1988), 513–522.

[73] G. Sherwood, Effective testing of factor combinations. In *Proc. of the Third Int. Conf. on Softw. Test., Anal., and Rev. (STAR94), Washington, D.C.*, Software Quality Eng., 1994.

[74] G. B. Sherwood, S. S. Martirosyan, and C. J. Colbourn, Covering arrays of higher strength from permutation vectors. *J. Comb. Des.* **14** (2006), 202–213.

[75] N. J. A. Sloane, Covering arrays and intersecting codes. *J. Comb. Des.* **1** (1993), no. 1, 51–63.

[76] J. Spencer, Nonconstructive methods in discrete mathematics. In *MAA Stud. in Math.*, vol. **17**, *Stud. in Comb.*, ed. G.-C. Rota, Math. Assoc. of America, 1978, pp. 142–178.

[77] E. Sperner, Ein Satz über Untermenge einer endlichen Menge. *Math. Z.* **27** (1928), 544–548.

[78] K. C. Tai and Y. Lei, A test generation strategy for pairwise testing. *IEEE Trans. on Softw. Eng.* **28** (2002), 109–111.

[79] D. T. Tang and C. L. Chen, Iterative exhaustive pattern generation for logic testing. *IBM J. Res. Dev.* **28** (1984), 212–219.

[80] D. T. Tang and L. S. Woo, Exhaustive test pattern generation with constant weight vectors. *IEEE Trans. Comput.* **32** (1983), 1145–1150.

[81] J. Torres-Jimenez, home page, with link to covering array repository:
`http://www.tamps.cinvestav.mx/∼jtj`.

[82] J. Torres-Jimenez and E. Rodriguez-Tello, Simulated annealing for constructing binary covering arrays of variable strength. *WCCI 2010, IEEE World Congress on Computational Intelligence, July, 18-23, Barcelona, Spain.*

[83] R. A. Walker II and C. J. Colbourn, Tabu search for covering arrays using permutation vectors. *J. Stat. Plan. and Inference* **139** (2009), 69–80.

[84] K. Yamamoto, Logarithmic order of free distributive lattices. *J. Math. Soc. Japan* **6** (1954), 343-353.

[85] J. Yan and J. Zhang, A backtracking search tool for constructing combinatorial test suites. *J. Syst. and Softw.* **81** (2008), 1681–1693.

[86] C. Yilmaz, M. B. Cohen, and A. Porter, Covering arrays for efficient fault characterization in complex configuration spaces. *ACM SIGSOFT Softw. Eng. Notes* **29** (2004), 45–54.

[87] Webpage for Automated and Combinatorial Testing of Software at the National Institute of Standards and Technology:
`http://csrc.nist.gov/groups/SNS/acts/index.html`.