THE FINAL DRAFT 18/7/03

Software and Hardware Testing Using Combinatorial Covering Suites

Alan Hartman
IBM Haifa Research Laboratory

Abstract: In the 21st century our society is becoming more and more dependent on software systems. The safety of these systems and the quality of our lives is increasingly dependent on the quality of such systems. A key element in the manufacture and quality assurance process in software engineering is the testing of software and hardware systems. The construction of efficient combinatorial covering suites has important applications in the testing of hardware and software. In this paper we define the general problem, discuss the lower bounds on the size of covering suites, and give a series of constructions that achieve these bounds asymptotically. These constructions include the use of finite field theory, extremal set theory, group theory, coding theory, combinatorial recursive techniques, and other areas of computer science and mathematics. The study of these combinatorial covering suites is a fascinating example of the interplay between pure mathematics and the applied problems generated by software and hardware engineers. The wide range of mathematical techniques used, and the often unexpected applications of combinatorial covering suites make for a rewarding study.

1. INTRODUCTION

Testing is an important but expensive part of the software and hardware development process. In order to test a large software or hardware system thoroughly, many sequences of possible inputs must be tried, and then the expected behavior of the system must be verified against the system's requirements. This is usually a labor-intensive process that requires a great deal of time and resources. It has often been estimated that testing consumes at least 50% of the cost of developing a new piece of software. The testing costs for hardware and safety-critical systems are often higher.

The consequences of inadequate testing can be catastrophic. An extreme example is the software failure in the Therac-5 radiation therapy machine [27] that is known to have caused six massive overdoses of radiation to be administered to cancer patients resulting in deaths and severe injuries. A further example of a catastrophic software failure occurred when the Ariane 5 satellite launcher exploded 40 seconds into its maiden flight. A register overflow failure which occurred simultaneously on one processor and on its backup [29] caused both processors to shut down, and eventually abort the satellite mission. The most celebrated hardware bug is the Pentium floating point division bug [14] which caused an error in the accuracy of a small number of division computations. This in itself, does not sound like a disaster – but the cost to the Intel corporation was measured in millions of dollars.

An approach to lowering the cost of software testing was put forward by Cohen, Dalal, Fredman, and Patton [11] using test suites generated from combinatorial designs. This approach involves identifying parameters that define the space of possible test scenarios, then selecting test scenarios in such a way as to cover all the pairwise (or *t*-wise) interactions between these parameters and their values. A similar approach was used earlier in hardware testing by Tang, Chen, and Woo [41,42] and Boroday and Grunskii [3]. The approach is familiar to statisticians, and has been used in the design of agricultural experiments since the 1940s [17]. The statistical analysis of such experiments is facilitated if every interaction is covered precisely the same number of times, however Cohen et al. point out that in software testing it is often sufficient to generate test suites so that each interaction is covered *at least* once rather than insisting on the more restrictive condition required by the statisticians.

As an example, consider the testing of an internet site that must function correctly on three operating systems (Windows, Linux, and Solaris), two browsers (Explorer and Netscape), three printers (Epson, HP, and IBM), and two communication protocols (Token Ring and Ethernet). Although there are $36 = 3 \times 2 \times 3 \times 2$ possible test configurations, the nine tests in Figure 1 cover all the pairwise interactions between different parameters of the system.

The interactions between operating systems and printers are all covered precisely once, but some interactions between operating systems and browsers are covered more than once. For example, Windows and Explorer are tested together twice in the test suite.

Operating System	Browser	Printer	Protocol
Windows	Explorer	Epson	Token Ring
Windows	Netscape	HP	Ethernet
Windows	Explorer	IBM	Ethernet
Linux	Netscape	Epson	Token Ring
Linux	Explorer	HP	Ethernet
Linux	Netscape	IBM	Token Ring
Solaris	Explorer	Epson	Ethernet
Solaris	Netscape	HP	Token Ring
Solaris	Explorer	IBM	Ethernet

Figure 1. A set of test cases with pairwise coverage

More generally, if a software system has k parameters, each of which must be tested with n_i values ($1 \le i \le k$), then the total number of possible test vectors is the product $\prod_i n_i$. If we wish to test the interactions of any subset of t parameters, then the number of test vectors may be as small as the product of the t largest values n_i.

The same argument applies to testing software that computes a function with k parameters, or a piece of hardware with k input ports. In the context of hardware testing it is of particular importance to find small sets of binary vectors of length k with the property that any fixed set of t coordinate places contains all 2^t binary strings of length t. In Figure 2 we illustrate a set of 8 binary vectors of length 4 such that any 3 coordinate places contain all possible binary strings of length 3.

In the next section we will formalize the problem of finding minimal covering suites. We then discuss various techniques for constructing good covering suites using finite fields (in Section 3), extremal set theory (in Section 4), group theory (Section 6), coding theory (Section 4), algorithmic methods (Sections 5 and 8), and combinatorial recursive methods (Section 7). We also discuss briefly the results on lower bounds on the sizes of covering suites in Section 4. Finally in Section 9 we close with an account of three diverse applications of covering suites including one in an area far removed from the original motivating problem.

```
0000
0011
0101
0110
1001
1010
1100
1111
```

Figure 2. A covering suite of strength 3 with four binary parameters

The wide range of methods used, and the variety of applications of these combinatorial objects provide evidence of the value of interdisciplinary studies, and the cross-fertilization that occurs between mathematics and computer science.

2. COVERING SUITES AND THEIR PROPERTIES

Let $D_1, D_2, ..., D_k$ be finite sets of cardinalities $n_1, n_2, ..., n_k$ respectively. A *test suite* with *N* test vectors is an array $A = (a_{ij} : 1 \le i \le N, 1 \le j \le k)$ where each member of the array $a_{ij} \in D_j$ for all *i* and *j*. The rows of the array are called *test vectors* or *test cases* or just simply *tests*. The set D_i is the domain of possible values for the *i*-th coordinate of the test vector.

We shall say that the test suite A is a *t-wise covering suite* with parameters $n_1, n_2, ..., n_k$ if for any *t* distinct columns $c_1, c_2, ..., c_t$ and for any ordered *t*-tuple $T \in D_{c_1} \times D_{c_2} \times ... \times D_{c_t}$ there exists at least one row r such that $(a_{rc_1}, a_{rc_2}, ..., a_{rc_t}) = T$.

We define the *covering suite number* $CS_t(n_1, n_2, ..., n_k)$ to be the minimum integer *N* such that there exists a t-wise covering suite with *N* test cases for k domains of sizes $n_1, n_2, ..., n_k$. The function is well-defined, since the actual members of the sets D_j are not important; what really matters is the cardinalities of the sets. Unless otherwise stated, we will assume that $D_j = \{0,1,...,n_j - 1\}$.

If all the domains are the same size, say n, we will denote $CS_t(n, n, ..., n)$ by $CS_t(n^k)$ and we also use this standard exponential notation for multi-sets in other contexts, so that for example, we will use $CS_t(n^2, m^3)$ for $CS_t(n, n, m, m, m)$.

A strict interpretation of the definition implies that $CS_0(n^k) = 1$, since at least one row is required to cover the empty 0-tuple. It is also straightforward to see that $CS_1(n^k) = n$, since each column of the minimal array must contain a permutation of I_n.

In the rest of this section, we will establish some elementary properties of covering suites and the covering suite numbers defined above.

Lemma 2.1: $CS_t(n_1, n_2, ..., n_k) \geq n_1 n_2 ... n_t$, and hence $n^k \geq CS_t(n^k) \geq n^t$

Proof: Consider the number of test cases required to cover all the combinations of values in the first *t* domains. Details are left as an exercise. □

We now show that $CS_t(n^k)$ is a non-decreasing function of t, n, and k.

Lemma 2.2: For all positive integer parameters, we have:

a) if $k < r$ then $CS_t(n^k) \leq CS_t(n^r)$

b) if $n_i \leq m_i$ for all *i* then $CS_t(n_1, n_2, ... n_k) \leq CS_t(m_1, m_2, ... m_k)$

c) if $n < m$ then $CS_t(n^k) < CS_t(m^k)$

d) if $s < t$ then $CS_s(n^k) \le CS_t(n^k)$.

Proof: a) Let $CS_t(n^r) = N$ and let A be a t-wise covering suite with *N* test cases for r domains of size n. Deleting $r-k$ columns from A leaves a *t*-wise covering test suite with *N* test cases for k domains of size n, and thus $CS_t(n^k) \le N = CS_t(n^r)$.

b) Let $CS_t(m_1, m_2, ..., m_k) = N$ and let A be a *t*-wise covering suite with *N* test cases for the k domains of size m_i. Replace every entry of A that lies in the set $I_{m_i} - I_{n_i}$ by an arbitrary member of I_{n_i} to produce a t-wise covering suite with *N* test cases for the k domains of size n_i, thus $CS_t(n_1, n_2, ..., n_k) \le N = CS_t(m_1, m_2, ..., m_k)$.

c) To prove the strict inequality in c), we observe that the symbols in any column may be permuted independently of each other without affecting the coverage property. We permute the symbols so that the first row of the larger array is the constant vector with value $m-1$ (the largest member of I_m). Now delete this row, and proceed as in the proof of part b).

d) This follows from the fact that every *s*-tuple is contained in some *t*-tuple. Moreover, the inequality is strict when $n > 1$ and $k \ge t$. □

The following result shows that there is a stronger relationship between the sizes of covering suites when increasing their strength *t.*

Lemma 2.3: We have $CS_t(n_1, n_2, ..., n_k) \ge n_1 CS_{t-1}(n_2, n_3, ..., n_k)$ and thus $CS_t(n^k) \ge nCS_{t-1}(n^{k-1})$.

Proof: Consider the n_1 sub-arrays of a *t*-wise covering array consisting of all rows where the first column takes a constant value, and delete the first column, see Figure 3. Each such sub-array must be a *(t*-1*)*-wise covering array, which implies the result. □

The problem of minimizing the number N of test cases in a t-wise covering test suite for k domains of size n was apparently first studied by Renyi [35], and many papers on the subject have appeared since then. Many of these consider the mathematically equivalent problem of maximizing the number k of domains of size n in a t-wise covering test suite with a fixed number N of test cases. This is known as the problem of finding the size of a largest *family of t-independent n-partitions of an N-set*. Other names used in the literature for test suites are *covering arrays, (k,t)-universal sets*, and *t-surjective arrays*.

0 0 0	$\geq CS_{t-1}(n_2, n_3, ..., n_k)$
1 1 1	$\geq CS_{t-1}(n_2, n_3, ..., n_k)$
. . .	$\geq CS_{t-1}(n_2, n_3, ..., n_k)$
$n_1 - 1$ $n_1 - 1$	$\geq CS_{t-1}(n_2, n_3, ..., n_k)$

Figure 3. Proof of Lemma 2.3.

3. ORTHOGONAL ARRAYS AND FINITE FIELDS

Orthogonal arrays are structures that have been used in the design of experiments for over 50 years. An orthogonal array of size N with k constraints, n levels, strength t, and index λ is an $N \times k$ array with entries from $I_n = \{0,1,...n-1\}$ with the property that: in every $N \times t$ submatrix, every $1 \times t$ row vector appears precisely $\lambda = N / n^t$ times. See Figure 4.

```
0000
0111
0222
1021
1102
1210
2012
2120
2201
```

Figure 4. An orthogonal array of strength 2.

In a fundamental paper, Bush [6] gave constructions for orthogonal arrays of index 1, and bounds on their parameters. It is clear that an orthogonal array of index 1 is a special case of a covering suite, since in a covering suite each $1 \times t$ row vector is required to appear *at least* once. Thus, an orthogonal array is always a minimal covering suite.

Orthogonal arrays of strength 2 and index 1 have been especially well studied as they are equivalent to mutually orthogonal Latin squares of order n.

A *Latin square* of *order* n is a square array of side n with entries from the set I_n with the property that every row and every column contains every member of the set precisely once. Two Latin squares of order n are said to be mutually orthogonal if for any ordered pair of elements $(x, y\} \in I_n^2$ there exists precisely one cell, such that the first square has the value x in the cell, and the second square has the value y in that cell. We illustrate two mutually orthogonal Latin squares of order 3 in

Figure 5 below. Notice that all nine pairs of symbols (x, y) occur once in the same cell of the squares.

$$\begin{array}{ccc} \mathbf{0} & \mathbf{1} & \mathbf{2} \\ \mathbf{2} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{2} & \mathbf{0} \end{array} \qquad \begin{array}{ccc} \mathbf{0} & \mathbf{1} & \mathbf{2} \\ \mathbf{1} & \mathbf{2} & \mathbf{0} \\ \mathbf{2} & \mathbf{0} & \mathbf{1} \end{array}$$

Figure 5. A pair of mutually orthogonal Latin squares of side 3.

A set of $k-2$ Latin squares of order n, each of which is orthogonal to the other, can be put in one-to-one correspondence with an orthogonal array of size n^2 with k constraints, n levels, strength 2, and index 1, as follows.

We define a row of the array for each of the n^2 cells in the Latin squares. The first column of each row contains the row number of the cell, the second column contains the column number of the cell, and the *j*-th column (for $j > 2$) contains the element in the cell of the *j*-2nd Latin square. We illustrate this construction using the two orthogonal Latin squares of order 3 given above to build the orthogonal array of size 9, with 4 constraints, 3 levels, strength 2, and index 1, given at the beginning of this section (Figure 4).

It is well-known (see, for example, [12]) that there exists a set of $n-1$ mutually orthogonal Latin squares of order n if and only if there exists a finite projective plane of order n, and that, moreover, the number of mutually orthogonal Latin squares of order n is at most $n-1$. We will see below that there are many values of n for which this holds, and they will help us to construct covering suites in many cases. We summarize this in the following result:

Theorem 3.1: $CS_2(n^k) = n^2$ for all $k \le n+1$ if and only if there exists a projective plane of order n, and $CS_2(n^k) > n^2$ for all $k > n+1$.

Proof: If there exists a projective plane of order n, then there exist $n-1$ mutually orthogonal Latin squares of order n, which implies the existence of an orthogonal array of strength 2, index 1, $n+1$ constraints, and n levels. Hence $CS_2(n^k) \le n^2$ for all $k \le n+1$, using the monotonicity results (Lemma 2.2). But, by Lemma 2.1, we have

$CS_2(n^k) \geq n^2$, and thus equality holds. On the other hand, if $CS_2(n^k) = n^2$, then each pair of symbols must occur together precisely once in each pair of columns, and hence the covering suite must be an orthogonal array, which in turn implies the existence of $k-2$ mutually orthogonal Latin squares, and hence $k \leq n+1$. □

It is also well known that projective planes exist for all orders $n = p^\alpha$, which are powers of a single prime p. The construction of projective planes of prime power order was generalized by Bush [6] who proved the following result.

Theorem 3.2: Let $n = p^\alpha$ be a prime power with $n > t$. Then $CS_t(n^k) = n^t$ for all $k \leq n+1$. Moreover, if $n \geq 4$ is a power of 2, then $CS_3(n^k) = n^3$ for all $k \leq n+2$.

Proof: Let $F = \{0,1,...\}$ be the set of elements in a field of order n, with 0 being the zero element of the field. We index the columns of the orthogonal array by members of $F \cup \{\infty\}$, and the rows of the array are indexed by *t*-tuples $(\beta_0, \beta_1, ..., \beta_{t-1}) \in F^t$.

The array is then defined by the following table:

Column	Row	Array Entry
∞	$(\beta_0, \beta_1, ..., \beta_{t-1})$	β_0
0	$(\beta_0, \beta_1, ..., \beta_{t-1})$	β_{t-1}
$x \neq 0$	$(\beta_0, \beta_1, ..., \beta_{t-1})$	$\sum_{j=0}^{t-1} \beta_j x^j$

Figure 6. Construction of orthogonal arrays of strength t

Let $T = (x_0, x_1, ..., x_{t-1})$ be a *t*-tuple of distinct columns, and let $B = (b_0, b_1, ..., b_{t-1})$ be an arbitrary member of F^t. To complete the proof of the first part of the theorem we need to show that B occurs as some row of the sub-matrix whose columns are indexed by T. If T contains neither ∞ nor 0, then we need to solve the following t equations for the t unknown quantities β_j, which index the row containing B.

$$\sum_{j=0}^{t-1} \beta_j x_i^{\ j} = b_i \text{ with } 0 \le i < t.$$

Now the coefficient matrix has the form of a Vandermonde matrix [43], and thus is invertible. Hence the system of equations has a unique solution. If T contains either ∞, or 0, or both, then we have a system of $t-1$ or $t-2$ equations that also have a Vandermonde coefficient matrix, and thus are uniquely solvable.

In the case where $t = 3$ and n is a power of two, we index the columns of the matrix by $F \cup \{\infty_0, \infty_1\}$ and we construct the array as follows:

Column	Row	Array Entry
∞_0	$(\beta_0, \beta_1, \beta_2)$	β_0
∞_1	$(\beta_0, \beta_1, \beta_2)$	β_1
0	$(\beta_0, \beta_1, \beta_2)$	β_2
$x \neq 0$	$(\beta_0, \beta_1, \beta_2)$	$\beta_0 + \beta_1 x + \beta_2 x^2$

Figure 7. Construction of orthogonal arrays of strength 3 over fields of characteristic 2.

The proof of this construction is similar, with the exception of the case in which we consider three columns indexed by two distinct non-zero field members, say x and y, and the column indexed by ∞_1. In this case we have to solve the following equations for β_0, β_1, and β_2.

$$\beta_0 + \beta_1 x + \beta_2 x^2 = b_0$$

$$\beta_0 + \beta_1 y + \beta_2 y^2 = b_1$$

$$\beta_1 = b_2$$

which reduces to the following pair of equations in β_0 and β_2:

$$\beta_0 + \beta_2 x^2 = b_0 - b_2 x$$

$$\beta_0 + \beta_2 y^2 = b_1 - b_2 y$$

Now these equations have a unique solution if and only if $x^2 - y^2 \neq 0$. In most fields this quantity may be zero for distinct values x and y, but in fields of characteristic 2, $x^2 - y^2 = (x-y)^2 \neq 0$. □

Remark 3.3: In the construction for arrays of strength 2, one can order the rows and columns so that the first q rows have the form $(0, x, x, ..., x)$ one for every member x of the field. This can be done by putting the 0 column on the left, and placing all rows where $\beta_1 = 0$ at the top of the array. (An example is shown in Figure 4.) Deleting these rows and the first column leaves us with an array with $q^2 - q$ rows, and q columns with the property that any ordered pair of distinct members of the field is contained in some row of any pair of columns. These structures are known in the literature as ordered designs (see Section IV.30 in [12]).We will use this construction in Section 7.

Bush [5] also gave the following product construction, which generalizes MacNeish's product construction for mutually orthogonal Latin squares.

Theorem 3.4: If there exist orthogonal arrays with k constraints, n_i levels (for $i = 1,2$), strength t, and index 1, then there exists an orthogonal array with k constraints, $n_1 n_2$ levels, strength t, and index 1.

Proof: The construction is a straightforward product construction, indexing the rows of the new array by ordered pairs of indices of the input arrays, and using the Cartesian product of the symbol sets used as the symbol set of the new array. If the two input arrays are $A[i, j]$ and

$B[m,j]$ then the resulting array $C[(i,m),j]$ is defined by $C[(i,m),j]=(A[i,j],B[m,j])$. The details are left as an exercise. □

Theorems 3.2 and 3.4 have the following consequences for covering suites:

Corollary 3.5: If $n=\prod q_j$ where the q_j are powers of distinct primes, then $CS_t(n^k)=n^t$, for any $k\le 1+\max(t,\min q_j)$.

There is a great deal of literature on the existence of sets of mutually orthogonal Latin squares; see [12] for an extensive list of references and numerical results. These results all have implications for the sizes $CS_2(n^k)$ of optimal pairwise covering suites with $k\le n+1$. We quote two of the most famous of these results – the disproof of Euler's conjecture and the Chowla, Erdös, Straus Theorem. We will use these results in Section 7.

Theorem 3.6 (Bose, Parker, and Shrikhande [4]): For any positive integer n other than 2 or 6, there exists a pair of mutually orthogonal Latin squares of side n, and thus $CS_2(n^4)=n^2$ for all $n\notin\{2,6\}$.

Euler originally conjectured (on the basis of Corollary 3.5) that no pair of mutually orthogonal Latin squares of side n exists for all $n\equiv 2(\mathrm{mod}\,4)$. A proof of the non-existence of a pair of squares of side 6 was published in 1900, but Parker eventually found a pair of squares of side 10, and soon after, in 1960, the conjecture was totally discredited by Theorem 3.6.

Theorem 3.7 (Chowla, Erdös, Straus [10]) The number of mutually orthogonal Latin squares of side n goes to infinity with n, and for sufficiently large n, that number is at least $n^{0.0675}$.

In other words, if we fix k, then for all sufficiently large n, $CS_2(n^k)=n^2$. For small values of k much better results than that provided by Theorem 3.7 are known. For example $CS_2(n^4)=n^2$ for all $n>6$. (Theorem 3.6), and $CS_2(n^5)=n^2$ for all $n>10$ (see [12]).

One other result that belongs in this section is a construction due to Stevens, Ling, and Mendelsohn [39]. They give a construction for near optimal covering suites using affine geometries over finite fields. It is one of the few constructions in the literature for covering suites where not all the domains are of the same size.

Theorem 3.8: Let $n = p^{\alpha}$ be a prime power then $CS_2((n+1)^1,(n-1)^{n+1}) \le n^2 - 1$.

The testing problem discussed in Section 1 is an application of the constructions described in this section. In that problem we wanted to test the pairwise interactions of a system with four parameters, two of cardinality 2, and two of cardinality 3. Now from Lemma 2.1 we have $CS_2(3^2,2^2) \ge 9$. From the construction in Theorem 3.1 (illustrated in Figure 4) we have $CS_2(3^4) \le 9$, and hence by the monotonicity results in Lemma 2.2, we can construct the test suite given in Figure 1, which illustrates that $CS_2(3^2,2^2) \le 9$. Thus the test suite in Figure 1 is optimal.

4. LOWER BOUNDS AND ASYMPTOTICS

Weak lower bounds on the function $CS_t(n^k)$ can be derived from non-existence theorems for orthogonal arrays. Two examples are presented below.

Theorem 4.1 (Bush): An orthogonal array with k constraints, n levels, strength t, and index 1 exists only if:

$$k \le \left\{ \begin{array}{lll} n+t-1 & if \quad n \equiv 0 (\mathrm{mod}\, 2) & and \quad t \le n \\ n+t-2 & if \quad n \equiv 1 (\mathrm{mod}\, 2) & and \quad 3 \le t \le n \\ \quad t+1 & if \qquad t \ge n & \end{array} \right\}$$

This implies that $CS_t(n^k) > n^t$ for all k greater than the bounds given in Theorem 4.1.

Some tighter bounds on the existence of sets of mutually orthogonal Latin square due to Metsch [29] are very complicated to state (see II.2.23 in [12]) but have the following implications on the size of covering suites when $n \equiv 1,2 (\mathrm{mod}\, 4)$ and $n < 100$.

Corollary to Metsch's Theorem: $CS_2(n^k) > n^2$

(i) for all $k > n-4$, when $n = 14,21,22$,

(ii) for all $k > n-5$, when $n = 30,33,38,42,46,54,57,62,66,69,70,77,78$, and

(iii) for all $k > n-6$, when $n = 86,93,94$.

These lower bounds are of little use in that they do not tell us how fast $CS_t(n^k)$ grows as a function of k. The only case where tight lower bounds have been proved is when $n = t = 2$. This result has been rediscovered several times (see Rényi[35], Katona[24], Kleitman and Spencer[25], and Chandra, Kou, Markowsky, and Zaks [7] for examples).

Theorem 4.2: For all $k > 1$ we have $CS_2(2^k) = N$ where N is the smallest integer such that

$$k \le \binom{N-1}{\lceil N/2 \rceil}$$

So we have the following table of exact values for $CS_2(2^k) = N$:

k	2-3	4	5-10	11-15	16-35	36-56	57-126
N	4	5	6	7	8	9	10

Figure 8. The precise values of $N = CS_2(2^k)$.

The proof of these lower bounds uses Sperner's lemma (see [22] for example) when *N* is even and the Erdös-Ko-Rado theorem [14] when *N* is odd.

The test suites that reach this bound may be constructed by taking the first test case to be the all 0 vector. The columns of the remaining

$N-1$ rows each contain precisely $\lceil N/2 \rceil$ ones, and each column is constructed by choosing a different $\lceil N/2 \rceil$-subset of the rows.

For example, when $n = t = 2$ and $k = 15$ we deduce that $N = 7$, and the optimal test suite is:

```
00000 00000 00000
11111 11111 00000
11111 10000 11110
11100 01110 11101
10011 01101 11011
01010 11011 10111
00101 10111 01111
```

Figure 9. An optimal pairwise covering suite for 15 binary parameters.

Gargano, Körner, and Vaccaro [20] established the following asymptotic results for the case of pairwise covering suites:

Theorem 4.3:

$$\lim_{k\to\infty} CS_2(n^k)/\log k = n/2$$

Theorem 4.3 was considerably strengthened in a subsequent paper [19] by the same authors.

Let E be a set of ordered pairs $E \subset I_n^2$. We can define $CS_E(n^k)$ to be the minimum number of rows in an array with k columns over I_n, with the property that every pair of columns contains each of the members of E in at least one of its rows. Theorem 4.3 deals with the case in which $E = I_n^2$ but the result was strengthened in [19] and shown to hold for any subset E that contains a perfect matching. The implications of this result on the size of test suites is that one doesn't gain a great deal by excluding some of the pairwise interactions between parameters from consideration when constructing the test suites.

Stevens, Moura, and Mendelsohn [39] have also proved some other lower bounds on the sizes of covering suites with $t = 2$. Many of their results are too complicated to state here, but their improvements on

Theorem 4.1 are easy to state, and provide useful information on small parameter sets.

Theorem 4.4: If $k \geq n+2$, and $n \geq 3$ then $CS_2(n^k) \geq n^2 + 3$ with the only exception being $CS_2(3^5) = 11$.

We turn our attention now to lower bounds when $t = 3$. Kleitman and Spencer [25] proved that $CS_3(2^k) \geq 3.212...\log k(1+o(1))$. I am not aware of any analogous results for $t > 3$.

The best asymptotic results on upper bounds for the covering suite numbers appear to come from coding theoretical constructions (e. g. Sloane [38]), and other deep results in probability theory and group theory.

A series of recent papers in the theoretical computer science community have been concerned with the de-randomization of randomized algorithms. As a by-product of this work, Naor and Naor [31], and Naor, Schulman and Srinvasan [32] have obtained results on binary covering suites ($n = 2$). Azar, Motwani, and Naor [1] have generalized these methods to apply to larger values of n. The best asymptotic upper bound given in these papers is:

Theorem 4.5 [32]: $CS_t(2^n) \leq 2^t t^{O(\log t)} \log n$

5. GREEDY COVERING SUITES AND PROBABALISTIC CONSTRUCTIONS

The first method that comes to a computer scientist's mind is to try to use a greedy algorithm to construct covering suites. The analysis of this algorithm gives a surprisingly good upper bound, but unfortunately, it is not a practical algorithm for the construction of good covering suites, since it has exponential complexity. The algorithm is as follows.

Let us define the *t-deficiency* $D_t(A)$ of a test suite A with k domains of size n to be the number of *t*-tuples of domain values not contained in any test case. Thus the deficiency of a covering suite is 0, whereas the deficiency of an empty set of test cases is

$$D(\phi) = \binom{k}{t} n^t$$

The greedy algorithm for the construction of a covering suite is to start with the empty test suite, $A_0 = \phi$, and at each stage to add a test case that decreases the deficiency by as much as possible. If A_S is the test suite after the choice of S test cases, then we will show that

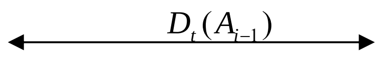
$$D_t(A_S) \le D_t(A_{S-1})(1 - n^{-t}) \qquad (1)$$

Thus $D_t(A_S) \le D_t(\phi)(1 - n^{-t})^S$. Let S be the smallest integer such that $D_t(A_S) < 1$ then $S = \left| -\log(D_t(\phi)/\log(1 - n^{-t}) \right|$ if we approximate $\log(1 - n^{-t})$ by $-n^{-t}$ we see that A_S is a covering suite when $S \approx n^t(\log\binom{k}{t} + t\log n)$ and hence we have:

Theorem 5.1: For all positive integers t, k, v, we have:

$$CS_t(n^k) \le n^t(\log\binom{k}{t} + t\log n)$$

Proof: To establish equation (1) let us consider the incidence matrix with n^k rows – one for each possible test case – and $\binom{k}{t} n^t$ columns – one for each *t*-tuple to be covered (see Figure 10). Each row of the matrix contains $\binom{k}{t}$ ones, and each column contains n^{k-t} ones. Let $m = D_t(A_i) - D_t(A_{i-1})$ be the maximum number of ones in a row of the submatrix indexed by the deficient columns (i.e., the columns of *t*-tuples not yet covered by the array A_{i-1}). Counting the number of ones in this submatrix in two ways (ones per row times number of rows, and ones per column times number of columns) we obtain the inequality $mn^k \ge D_t(A_{i-1})n^{k-t}$, which implies equation (1). □



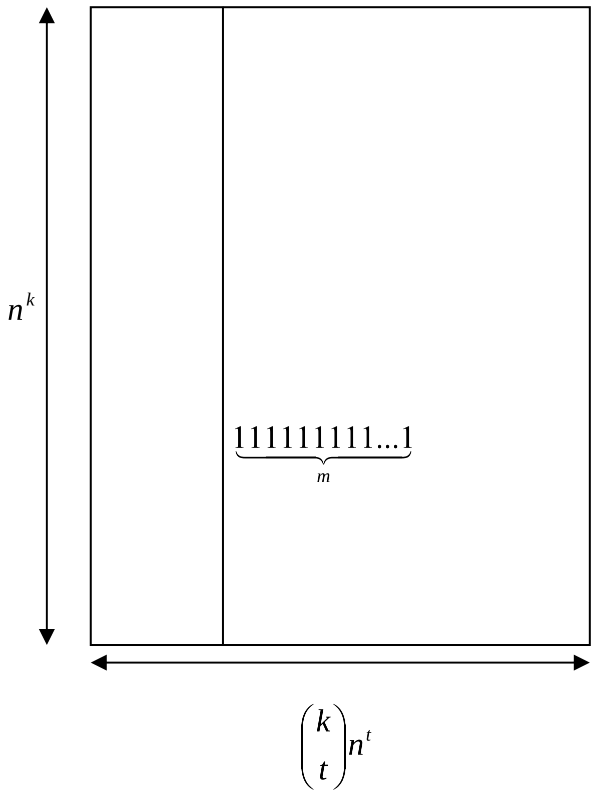


Figure 10. Proof of Equation (1).

The major problem with this construction is that in order to compute the next test case, one must consider all $n^k - S$ possible test cases in order to choose one that decreases the deficiency as much as possible.

Godbole, Skipper, and Sunley [21] use probabilistic arguments to show that a randomly chosen array, in which each symbol is selected with equal probability, has a positive probability of being a covering suite if the number of rows is large enough. Their result is stated below in Theorem 5.2:

Theorem 5.2: As $k \to \infty$,

$$CS_t(n^k) \leq \frac{(t-1)\log k}{\log(n^t/(n^t-1))}\{1+o(1)\}$$

Both Sloane[38] and Godbole et. al. [21] also prove the following stronger result due to Roux [36] for the case in which $t = 3$ and $n = 2$. The bound is stronger because the construction considers only the selection of columns with an equal number of zeros and ones, rather than selecting at random each entry in the matrix with a probability of ½.

Theorem 5.3: As $k \to \infty$,

$$CS_3(2^k) \le 7.56444 \ldots \log k\{1 + o(1)\}$$

6. ALGEBRAIC CONSTRUCTIONS

Two papers by Chateauneuf, Colbourn and Kreher [9] and Chateauneuf and Kreher [8] illustrate methods of using algebraic techniques to construct covering suites.

Let S be an $N \times k$ array over the set I_n. Let Γ be a subgroup of the group of all permutations of the symbols in I_n, and let $m = |\Gamma|$. For $g \in \Gamma$, we define the image Sg of S to be the array whose entries are the images of the corresponding members of S under the action of the permutation g. We further define the image S^Γ of S under Γ to be the $mN \times k$ array consisting of the rows of Sg for all $g \in \Gamma$.

For example, let $\Gamma = \{(0)(1)(2), (012), (021)\}$ be the cyclic group of permutations of I_3, and let

$$S = \begin{matrix} 000 \\ 012 \\ 021 \end{matrix}$$

Then S^Γ has nine rows, the first three being the image of S under (0)(1)(2), the identity permutation; the next three are the image of S under (012); and the last three are the image of S under (021).

$$S^\Gamma = \begin{array}{c} 000 \\ 012 \\ 021 \\ 111 \\ 120 \\ 102 \\ 222 \\ 201 \\ 210 \end{array}$$

The array S is called a starter array with respect to Γ if every $n \times t$ subarray contains at least one representative of each orbit of Γ acting on ordered *t*-tuples from I_n.

In the previous example, there are three orbits of ordered pairs under Γ - the orbit of 00, which contains the pairs $\{00, 11, 22\}$, the orbit of 01, which contains the pairs $\{01, 12, 20\}$ and the orbit of 02, which contains the pairs $\{02, 10, 21\}$. The array S is a starter array with respect to Γ since every 3×2 subarray contains a representative of each orbit of ordered pairs, and thus S^Γ is a $CS_2(3^3)$.

Note that we may also reduce the number of rows in S^Γ by deleting a copy of any row that is repeated.

For example, with $t = n = 3$, and Γ is the symmetric group of all six permutations of I_3, there are five orbits of ordered triples, namely the orbits of 000, 001, 010, 011, and 012. The array

$$S = \begin{array}{c} 0000 \\ 0012 \\ 0101 \\ 0110 \\ 0122 \end{array}$$

contains representatives of each of the five orbits in each of the four 3x5 subarrays. Taking the image of S under the symmetric group and removing the duplicate images of the first row generates an array with 27 rows, which establishes that $CS_3(3^4) = 27$ (c.f., Theorem 3.2).

The main constructions given in [9] involve the use of the projective general linear group $PGL(q)$ defined as a group of permutations of the projective line $GF(q) \cup \{\infty\}$, where q is a prime power and $GF(q)$ is the Galois field of order q. The projective linear group consists of all permutations of $GF(q) \cup \{\infty\}$ in the set

$$PGL(q) = \left\{ x \mapsto \frac{ax+b}{cx+d} : a,b,c,d \in GF(q), ad - bc \neq 0 \right\}$$

where we define $1/0 = \infty, 1/\infty = 0, \infty + 1 = 1 - \infty = 1 \times \infty = \infty,$ and $\infty / \infty = 1$.

The size of the group is $q^3 - q$. The action of this group on the projective line is sharply 3-transitive, meaning that there is a unique member of the group that takes any ordered triple of distinct elements to any other ordered triple of distinct elements. This means that if an array S over $GF(q) \cup \{\infty\}$ has the property that any $N \times 3$ subarray contains the five rows with patterns (xxx), (xxy), (xyx), (xyy), and (xyz), then the image of S with duplicates removed is a covering suite of strength 3.

Chateauneuf et al. 9] then give an ingenious construction of an array with $2n-1$ rows and $2n$ columns over I_n with the property that every $(2n-1) \times 3$ subarray contains representatives of the four non-constant orbits of triples (xxy), (xyx), (xyy), and (xyz). Taking the image of this array under $PGL(q)$, where $q = n-1$, together with the n constant rows of length, $2n$ gives a covering suite of strength 3. A special case of their construction of the starter array is given by indexing the rows by I_{2n-1}, the columns by I_{2n}, and setting $S[i, 2n-1] = 0$ and $S[i, j] = \min\{|i-j|, 2n-1-|i-j|\}$ for all $i, j \in I_{2n-1}$. This array S has the property that for any two columns j_1 and j_2 there is precisely one row i, such that $S[i, j_1] = S[i, j_2]$ and, furthermore, all other entries in that row are different from the common value. The exact value of i is given by solving the equation $2i = j_1 + j_2 \pmod{2n-1}$ or $i = j_1$ in the special case that $j_2 = 2n-1$. We give an example of this construction in Figure 11.

	0	**1**	**2**	**3**	**4**	**5**
0	0	1	2	2	1	0
1	1	0	1	2	2	0
2	2	1	0	1	2	0
3	2	2	1	0	1	0
4	1	2	2	1	0	0

Figure 11. A starter array for $PGL(3)$.

This property also ensures that given any three columns, there are precisely three rows in which not all of the entries in the row are distinct. Thus, so long as $2n-1>3$, we have a representative of the orbit of (xyz) in some row. Hence, we have the following:

Theorem 6.1: Let q be a prime power, then

$$CS_3((q+1)^{2(q+1)}) \leq (2q+1)(q^3-q)+q+1$$

A consequence of this result is that $CS_3(3^6) = 33$. The theorem provides the upper bound; the lower bound is a consequence of Lemma 2.3 and the result that $CS_2(3^5) = 11$, see [38].

There is another algebraic starter construction in [9] which establishes that $CS_3(4^8) \leq 88$.

7. RECURSIVE CONSTRUCTIONS

A recursive construction for a covering suite is a method for constructing a covering suite from one or more covering suites with smaller parameter sets. We begin with an efficient recursive construction for pairwise covering suites. This construction was probably known to Cohen et al.[11] – since the array they construct to show that $CS_2(3^{13}) \leq 15$ has the form of the construction. However, it first appeared in full detail in Williams' paper [45].

Theorem 7.1: If q is prime power, then $CS_2(q^{kq+1}) \leq CS_2(q^k) + q^2 - q$.

Proof: Let A be a pair-wise covering test suite with k columns and $N = CS_2(q^k)$ rows with entries from I_q.

Construct a new array B with N rows and $kq+1$ columns by taking each column A^i of A q times and bordering it with an additional column of zeroes. The additional $q^2 - q$ rows are constructed by taking the last rows from the orthogonal array C constructed in the proof of Theorem 3.2 and Remark 3.3, taking the first column once, and the remaining q columns k times each (see Figure 12). □

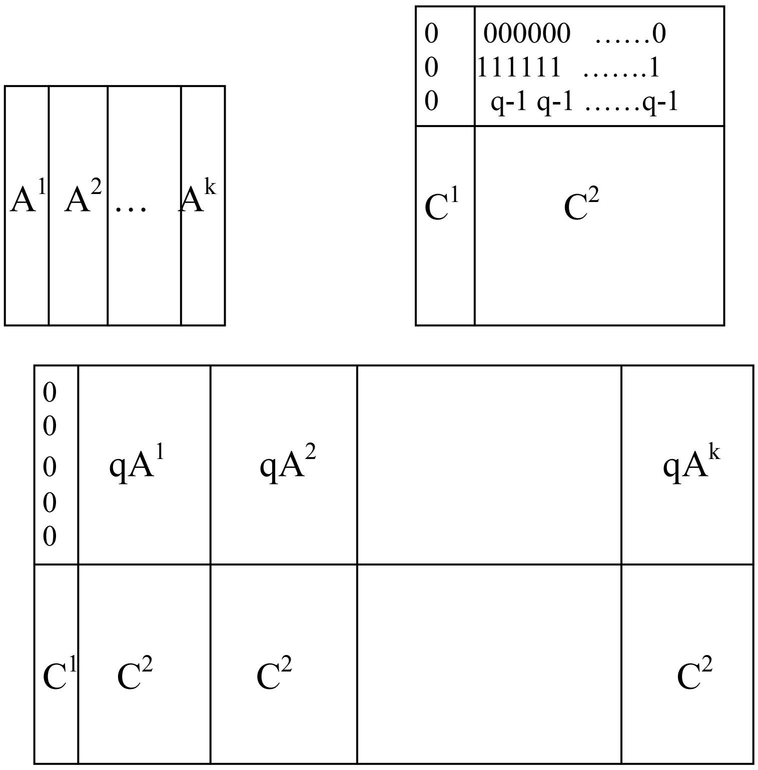


Figure 12. Construction for Theorem 7.1.

Corollary 7.2: If q is prime power, and d is any positive integer, then $CS_2(q^{q^d+q^{d-1}+\ldots+q+1}) \le dq^2-(d-1)q$.

Proof: The result follows from Theorem 3.2 when $d=1$, and by induction on d using Theorem 7.1.□

This result implies a good constructive upper bound on the size of pairwise covering test suites.

Theorem 7.3: There is an absolute constant C such that $CS_2(n^k) \le Cn^2 \log k$ for all positive integers k and n.

Proof: By Bertrand's postulate (proved by Chebyshev in 1851) there is a prime p, between n and $2n$. In fact, for $n>115$, there is always a prime between n and $1.1n$ (see [23]). Let d be the smallest integer such that $k \le 1 + p + p^2 + \ldots + p^d$.

This implies that $1 + p + p^2 + \ldots + p^{d-1} < k$, and hence that $d = O(\log k)$. Now applying Corollary 7.2 and the monotonicity result, Lemma 2.2, we have:
$CS_2(n^k) \le CS_2(p^{1+p+\ldots+p^d}) \le dp^2 = O(n^2 \log k)$ thus proving the theorem.□

Another recursive construction, which has been rediscovered many times is the following result, which gives a method of squaring the number k of parameters in a covering suite of strength t while multiplying the number of test cases by a factor dependent only on t and n, but independent of k. This factor is related to the Turan numbers $T(t,n)$ (see [44]) that are defined to be the number of edges in the Turan graph. The Turan graph is the complete *n*-partite graph with t vertices, having b parts of size $a+1$, and $n-b$ parts of size $a = \lfloor t/n \rfloor$ where $b = t - na$. Turan's theorem (1941) states that among all *t*-vertex graphs with no $n+1$ cliques, the Turan graph is the one with the most edges.

Note that when $n \ge t$, $T(t,n) = t(t-1)/2$, and that when $n = 2$, we have $T(t,2) = \lfloor t^2/4 \rfloor$.

Theorem 7.4: If $CS_t(n^k) = N$ and there exist $T(t,n) - 1$ mutually orthogonal Latin squares of side k (or equivalently $CS_2(k^{T(t,n)+1}) = k^2$) then $CS_t(n^{k^2}) \le N(T(t,n)+1)$.

Before proving this result, we note that this generalizes Tang and Chen's result [41], since they require k to be a prime. It generalizes and strengthens the result of Chateuneuf et al. [9] by removing the divisibility conditions on k, and producing smaller arrays in the cases where $n < t$.

Proof: Let A be a *t*-wise covering test suite with k columns, N rows, entries from I_n, and let A^i be the *i*-th column of A. Let $B = B[i, j]$ be an orthogonal array of strength 2 with $T(t,n)+1$ columns and entries from $\{1,2,...,k\}$. We will construct a block array C with k^2 columns and $T(t,n)+1$ rows. Each element in C will be a column of A. Let $A^{B[j,i]}$ be the block in the *i*-th row and *j*-th column of C (see Figure 13).

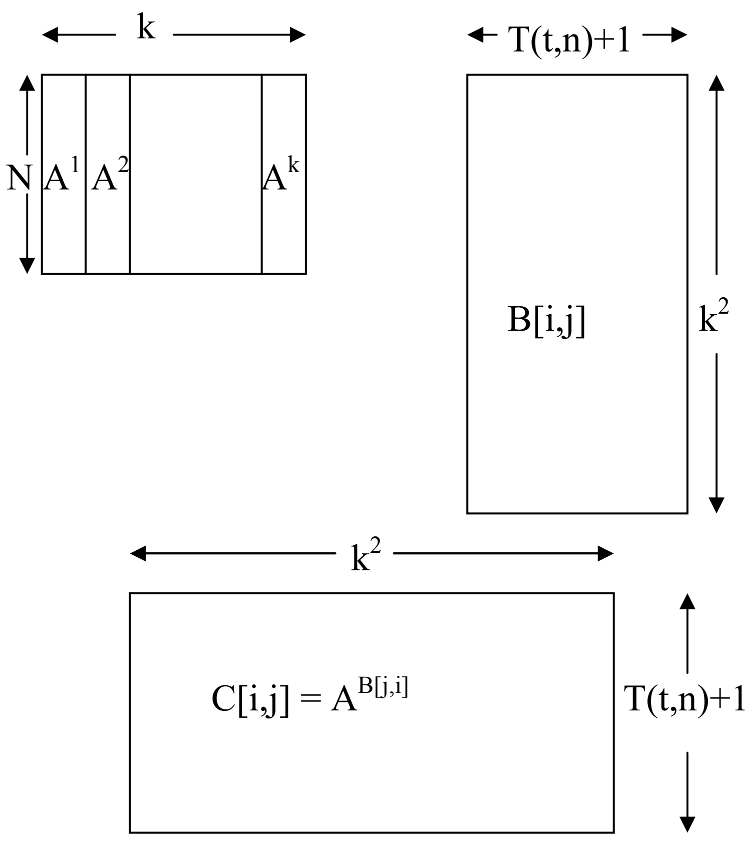


Figure 13. The construction for Theorem 7.4.

Now consider *T,* an arbitrary *t*-tuple of members of I_n. Let C' be a submatrix of C induced by an arbitrary choice of t columns. We wish to show that *T* is a row of C'.

The columns of C' correspond to t rows of B. Let B' be the submatrix of B induced by those t rows. We wish to find a column in B' with distinct values whenever *T* has distinct values in the corresponding coordinates, since this would guarantee that *T* is a row in C' using the properties of the base array A.

Since B is an orthogonal array, whenever $B[i, j] = B[k, j]$ then $B[i,m] \neq B[k,m]$ for every column $m \neq j$. This means that any pair of distinct values in *T* eliminates at most one column of B'. By Turan's theorem, the number of pairs of distinct values in *T* is at most $T(t,n)$, and hence at least one column in B' contains distinct values whenever *T* contains distinct values. This completes the proof. □

The result has several immediate corollaries.

Corollary 7.5 (Boroday [1]): $CS_3(2^{k^2}) \leq 3CS_3(2^k)$.

Proof: The result follows from Theorem 7.4 since $T(3,2) = 2$ and $CS_2(k^3) = k^2$ for all k, by Corollary 3.5. □

Using the disproof of Euler's conjecture we can derive:

Corollary 7.6: For all $k > 2,\ k \neq 6$, $n > 2$ we have $CS_3(n^{k^2}) \leq 4CS_3(n^k)$.

Proof: This result also follows from Theorem 7.4 since $T(3,n) = 3$ for all $n > 2$, and $CS_2(k^4) = k^2$ for all $k > 2,\ k \neq 6$, by Theorem 3.6. □

We also derive one more corollary using the result of Chowla, Erdös, and Strauss:

Corollary 7.7: For all positive integers t and n, and all sufficiently large integers k, we have $CS_t(n^{k^2}) \leq (T(t,n)+1)CS_t(n^k)$.

We give one more recursive result that shows how to double the number of columns in 3-wise and 4-wise covering suites. The result for $t = 3$ and $n = 2$ was first proved by Roux [36]. The result for $t = 3$ appears in [9], although our proof is different. To the best of my knowledge, the result for $t = 4$ is new.

A tool used in the construction is a partition of the edge set of the complete directed graph on *n* vertices, such that each part contains a spanning set of directed cycles – i.e., each vertex occurs precisely once as the head of an arc and once as the tail of an arc. The simplest such partition is given by the following construction:

$$F_j = \{(i, i + j(\bmod n)) : i \in I_n\}, j = 1,2,...,n-1$$

It is clear from the construction that each vertex appears precisely once as the head of an arc in F_j, and precisely once as the tail of some arc. To show that each arc in the complete directed graph occurs in precisely one of the sets F_j, consider the arc (x, y) and note that it occurs in the set F_{y-x}.

We now proceed to use this construction as an ingredient in the following doubling constructions for strength 3 and 4 covering suites.

Theorem 7.8: For all positive integers n and k,

a) $CS_3(n^{2k}) \le CS_3(n^k) + (n-1)CS_2(n^k)$

b) $CS_4(n^{2k}) \le CS_4(n^k) + (n-1)CS_3(n^k) + CS_2((n^2)^k)$

Proof: a) Let A be a 3-wise covering suite with k columns and $CS_3(n^k)$ rows over the symbol set I_n. Construct a new array by taking each column of A twice and adding $(n-1)CS_2(n^k)$ rows constructed as follows. Let B be a pairwise covering suite with k columns and $CS_2(n^k)$ rows over the symbol set I_n. Take $n-1$ copies of B, and replace the *i*-th symbol in the *j*-th copy with the *i*-th member (an ordered pair) of F_j. A worked example of this construction is given below.

To verify that this construction yields a 3-wise covering suite, we need to verify that three types of triples are covered by some row of the array: triples with three elements in distinct columns of the original array A, triples with two equal symbols in columns that came from the same column of A, and triples with two unequal symbols in columns that came from the same column in A. The first and second types of triples are covered due to the original structure of A, and the third type of triple is covered by the construction of the additional rows from B.

The construction and proof for b) is similar. Let A, B, and C be 4-wise, 3-wise, and pairwise covering suites with the parameters given in the statement of the result. Take each column of A twice, take $n-1$ copies of B and replace the *i*-th symbol in the *j*-th copy with the *i*-th member of F_j, then take a copy of C, and replace the *i*-th symbol with the *i*-th member of $I_n \times I_n$ in some arbitrary ordering of the members of this Cartesian product. □

Example 7.6: We now illustrate the use of Theorem 7.5 to show that:

$$CS_3(3^8) \leq 45 = 27 + 9 + 9$$

The first 27 rows of the array come from doubling each column of the array constructed in Theorem 3.2, which shows that $CS_3(3^4) = 27$.

```
0000  →   00 00 00 00
0012  →   00 00 11 22
0021  →   00 00 22 11
            ...
2212  →   22 22 11 22
2220  →   22 22 22 00
```

Figure 14. The doubling process in the construction of Theorem 7.5.

The next 9 rows come from substituting the members of $F_1 = \{01,12,20\}$ for the three symbols in Figure 4, which is a minimal $CS_2(3^4)$, see Figure 15.

The final 9 rows come from substituting the members of $F_2 = \{02,10,21\}$ in the same array. □

Results similar to those of Theorem 7.5 are probably true for higher values of t, but the number of cases in the proof detracts from the aesthetics of such results.

0000	→	01 01 01 01
0111	→	01 12 12 12
0222	→	01 20 20 20
1021	→	12 01 20 12
1102	→	12 12 01 20
1210	→	12 20 12 01
2012	→	20 01 12 20
2120	→	20 12 20 01
2201	→	20 20 01 12

Figure 15. The substitution process in the construction of Theorem 7.5.

8. HEURISTICS

In this section, we discuss how the techniques presented in the previous sections can be used and extended by heuristic methods to solve practical problems in the generation of covering suites.

The most common problem, and the one on which we have focused so far, is that of generating a minimal set of test cases guaranteeing *t*-wise coverage of k parameters with domains of sizes $n_1, n_2, ..., n_k$. The practical issues of limited time and space require that we should find a polynomial time algorithm to solve this problem. Lei and Tai [26] prove that the determination of $CS_2(n^k)$ is NP-complete using a reduction to the vertex cover problem. Seroussi and Bshouty [37] prove that the determination of $CS_t(2^k)$ is NP-complete using a reduction to graph 3-coloring. Thus, it seems unlikely that we will find a polynomial time algorithm for constructing minimal covering suites in the general case.

Another interesting and practical problem is that of finding a test suite with minimal deficiency, given a fixed budget for executing a maximum of N test cases. This problem is theoretically equivalent to the problem of finding a minimal test suite, so it, too, is NP-complete.

Yet a third problem is that of finding a minimal test suite with a fixed *relative deficiency*, where the relative deficiency is defined as the deficiency divided by the total number of t-subsets to be covered. In the case where all domains are the same size, the relative deficiency RD_t of a test suite A is defined as:

$$RD_t(A) = \frac{D_t(A)}{n^t \binom{k}{t}}$$

A surprising result from Roux's thesis [36] states that for any t and any $\varepsilon > 0$, there is a constant $N(t,\varepsilon)$, independent of k, such that there exists a test suite A with $N(t,\varepsilon)$ test cases for k binary parameters with relative deficiency $\varepsilon = RD_t(A)$. Sloane's paper [38] quotes the result that $N(3,0.001) \leq 68$. Unfortunately, the arguments are probabilistic, and they do not give a deterministic algorithm for finding such a test suite.

Lei and Tai [26] also discuss the practical issue of extending a given test suite. Assuming that a pairwise covering test suite for k parameters is already given, what is the best way to add a single column, and perhaps additional rows, in order to extend this suite for the additional parameter? They give an optimal algorithm for adding new rows once a single column has been added to the initial test suite. However, their algorithms for adding a new column are either exponential or sub-optimal.

Our heuristics for solving these problems are a combination of the constructive and recursive methods given in sections 3 and 7, probabilistic algorithms inspired by Roux's techniques, and a greedy heuristic for the completion of partial test suites. We also use the monotonicity results when the domain sizes are inappropriate for the finite field methods of Section 3.

Roux's technique is particularly appropriate in the case of a fixed testing budget N. To apply the technique, we generate k random sets of columns of length N, with each symbol appearing either $\lfloor N/n_i \rfloor$ or $\lceil N/n_i \rceil$ times in the column. We then select one column from each of these sets in such a way as to minimize the deficiency of the array that we generate. We use a greedy heuristic for the minimization, since the selection of columns is reducible to the problem of finding a maximal clique in a graph.

Our heuristic for the completion of a partial test suite is different from that given by Cohen, Dalal, Fredman, and Patton in [11]. Assume that we are given a partial test suite, and we are required to add a new test case. We first find the set of t columns with the largest number of missing *t*-tuples, and select one of the missing tuples as the values in those columns. We then rank all the remaining (column, value) pairs by computing *t* values, ($p_0, p_1, \ldots p_{t-1}$) - which we call the *potential vector*. The first of these values p_0 is the amount by which the inclusion of the ranked value in the ranked column in the partial test case would decrease the deficiency. In other words, p_0 counts the number of *t*-tuples containing the value in the column and *t*-1 other values that have already been fixed in the partial test case under construction. In general, p_i counts the total number of missing *t*-tuples containing that value in that column as well as $t-1-i$ values that have already been fixed, and i undecided values in the other columns. We then choose the (column, value) pair with the lexicographically maximum potential vector. If several pairs achieve the same maximum potential vector, we break the tie by a random choice among those pairs that achieve the maximum.

For small values of t, k , and n , Nurmela [33] has used the method of tabu search effectively to find small covering suites, and some of the smallest known suites according to the tables in [9] are due to this technique.

9. APPLICATIONS

In the introduction we discussed the application of covering suites to the testing of a software or hardware interface with k parameters. In this section we discuss three other applications of covering suites: two in the area of software and hardware testing, and one in the seemingly unrelated area of search algorithms.

9.1 Reducing State Machine Models

It is common in both the hardware and software domains to specify the behavior of a unit to be tested by a state machine or transition system. A full account of this area may be found in [34]. The states represent the possible states for a software unit (e.g., the screen currently displayed, and the fields and actions currently active or disabled). The transitions between states are arcs labeled by the input stimulus from the user that cause the unit to change state. A great deal of research activity has been devoted to the analysis of these state machine models, but most of this effort is stymied by the so-called state explosion problem. The size of the state machine grows exponentially, even when the unit under test has a relatively simple structure.

In [16] Farchi, Hartman, and Pinter describe the use of a state machine exploration tool to generate test cases for the testing of implementations of software standards. In this context, a test case is not merely a *k*-tuple of input values, but it is a sequence of input stimuli, each of which is a tuple of input values. In graph theoretical terms, it is a path or walk through the arcs of the graph of the state machine.

The test generation tool described in [16] is built on the basis of the Murφ model checker [13]. The Murφ system builds a labeled directed graph to describe all possible behaviors of the system under test. The number of arcs leaving any particular state is equal to the number of possible stimuli which the tester can apply to the system at that state. Since the graph describes all possible behaviors, the number of arcs leaving any state is equal to the cardinality of the Cartesian product of the domains of all the input parameters. If instead of building an arc for each member of the Cartesian product, we use a covering suite subset of those arcs, we obtain a much smaller state machine. The new state machine does not describe all possible behaviors of the system under test, but it provides a good sample of the full set of behaviors. In model checking it is vital to describe the entire state machine, but in the context of test generation it is sufficient to sample the behavior of the system under test.

An example of the use of this technique is provided by the model we used to test the file system described in [16]. The model of the file system contains two users who interact with the system by sending requests to read, write, open, or close a file. The open and close commands do not have parameters, but the read and write commands each have five parameters in the model, two with domains of size 4, two of size 3, and one binary parameter. The states of the model may be divided into three classes, states where neither user may issue a read or write command, states where only one of the users may issue a read or write command, and states where both users may issue these commands.

In the full Cartesian product model, each state of the second class has $288 = 4 \times 4 \times 3 \times 3 \times 2$ "read" arcs leaving it, and the same number of "write" arcs. States of the third class have twice that many arcs leaving them since both users can perform both reads and writes. We can create a covering suite with only 16 rows which will cover all pairwise interactions between the parameters of the read and write commands. This reduces the number of arcs leaving a state of the third class from$1152 = 288 \times 4$ to $64 = 16 \times 4$.

The resulting state machine has fewer reachable states, and thus does not describe the full range of file system behavior, but it is adequate for generating a comprehensive test suite which ensures compliance to the software standard.

9.2 Path Coverage Using Covering Suites

A further application of covering arrays in the context of state machine models is their use as a means of achieving a reduced form of path coverage of the model.

When a state machine model is used to generate test cases for an application, the most common criteria for generating the test suite is the achievement of a coverage goal. A suite of test cases that passes once through every state is said to achieve *state coverage.* In graph theory terms, this resembles a path partition of the underlying directed graph. In conformance testing it is common practice to generate an Euler tour of the transition system to guarantee transition coverage. A stronger form of coverage requires the coverage of all paths of length two or greater.

We have used covering suites to define a test generation strategy that gives a moderate sized test suite with interesting path coverage properties. Recall that in the state machine, each arc is labeled by the stimulus that causes the unit under test to change state. We can view test cases with k transitions as sequences of arc labels, and if we select sequences of arc labels from a covering suite of strength 2, we achieve a level of coverage guaranteeing that all pairs of transition types are tested at all distances of up to k apart.

In the file system example discussed in the previous section we may wish to generate a series of test cases each containing 10 read commands, alternating between the two users. Having reduced the number of possible read commands to 16, we are still faced with the prospect that there are 16^{10} possible test cases with 10 steps. Again we can restrict ourselves to a set of sequences of length 10 taken from a pairwise covering suite, and with these 256 test cases we guarantee that all of the pairwise interactions between the various read commands have been tested together in the same test sequence.

9.3 Blind Dyslectic Synchronized Robots on a Line

A final application of covering suites, in a completely different area was pointed out to us by S. Gal.

Lim and Alpern [27] have studied the minimax rendezvous time problem for k distinguishable robots on a line. The robots are initially placed at the points 1, 2,…, k in some permutation. The robots are dyslectic in the sense that they do not have a common notion of the positive direction on the line. The robots cannot see each other and they only become aware of each other's presence by the sense of touch. The minimax rendezvous time is the minimum over all strategies of the maximum over all possible starting configurations of the time by which they can all get together. All the robots move at the same speed and are synchronized to start their rendezvous strategy together.

Gal[18] has exhibited a simple strategy where, in the first phase, the robots at positions 1 and k identify the fact that they are at the extreme positions, and then, in a second phase proceed toward the center, gathering all their colleagues, who remain passive in the second phase. In Gal's algorithm the first phase takes $2(1+\lceil \log_2 k \rceil)$ steps, and the second phase takes $\lceil k/2 \rceil$ steps. We propose a variant on Gal's algorithm, which decreases the number of steps required in the first phase.

We provide each robot with a binary sequence of length $N = CS_2(2^k)$. The sequence for the *k*-th robot is the *k*-th column of an optimal covering suite. The robot interprets a 0 as an instruction to move half a unit to its left, then half a unit right; a 1 is interpreted as a move half right then half left. Since the robots do not have a common sense of direction, in order to guarantee that each internal robot meets both its neighbors, the pair of robots must execute the four patterns 00, 01, 10, and 11. After precisely N time units, all the internal robots have met both their neighbors, and thus the end robots can be sure that they are on the ends, and begin the second phase. Theorem 4.2 guarantees that $N < 1 + \lceil \log_2 k \rceil$.

FURTHER READING

In this section we give a series of urls for web based resources relevant to the subjects discussed in this paper.

Thomas Huckle has a very interesting collection of software bugs at http://wwwzenger.informatik.tu-muenchen.de/persons/huckle/bugse.html

Peter Cameron's "Design Resources on the Web": http://www.maths.qmw.ac.uk/~pjc/design/resources.html

Neal Sloane's library of orthogonal arrays is at : http://www.research.att.com/~njas/oadir/index.html

The Handbook of Combinatorial Designs [12] has its own web page which is regularly updated with new results: http://www.emba.uvm.edu/~dinitz/hcd.html

Telecordia's web-based service for the generation of test cases is at: http://aetgweb.argreenhouse.com/

Harry Robinson's Model-based testing website has many interesting resources: http://www.geocities.com/model_based_testing/

A search at http://citeseer.nj.nec.com with the keyword "derandomization" yields a wealth of material on the asymptotic results quoted in Section 4.

ACKNOWLEDGMENTS

I would like to thank Marty Golumbic, Cindy Eisner, and Shmuel Gal for reading earlier drafts of this paper and providing helpful insights. I would like to thank Leonid Raskin who assisted in the implementation of the constructive algorithms and heuristics. I would also like to thank Gyorgy Turan and Seffi Naor for their help with the recent work on the de-randomization of algorithms. I would also like to thank Serge Boroday for his helpful correspondence, and Brett Stevens for his careful reading of the paper and for pointing out an error in an earlier proof of Theorem 7.8.

REFERENCES

1. Azar J., Motwani R., and Naor J., Approximating probability distributions using small sample spaces. Combinatorica 18 (1998) 151-171.

2. Boroday S. Y.***, Determining essential arguments of Boolean functions***. (in Russian), in Proceedings of the Conference on Industrial Mathematics, Taganrog 1998, 59-61. The translation is a personal communication from the author.

3. Boroday S. Y. and Grunskii I. S., ***Recursive generation of locally complete tests.*** Cybernetics and Systems Analysis 28 (1992), 20-25.

4. Bose R. C., Parker E. T., and Shrikhande S., ***Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler's conjecture***. Can. J. Math. 12(1960) 189-203.

5. Bush K. A., ***A generalization of the theorem due to MacNeish***. Ann. Math. Stat. 23 (1952) 293-295.

6. Bush K. A., ***Orthogonal arrays of index unity.*** Annals of Mathematical Statistics 23 (1952) 426-434.

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

8. Chateauneuf M. A. and Kreher D. L., ***On the state of strength 3 covering arrays.*** J. Combinatorial Designs, to appear. Available at http://www.math.mtu.edu/~kreher/.

9. Chateauneuf M. A., Colbourn C. J., and Kreher D. L., ***Covering arrays of strength 3***. Designs, Codes, and Cryptography, 16 (1999) 235-242.

10. Chowla S., Erdös P., and Straus E. G., ***On the maximal number of pairwise orthogonal Latin squares of a given order***. Canad. J. Math. 13 (1960) 204-208.

11. Cohen D. M., Dalal S. R., Fredman M. L., and Patton G. C., ***The AETG System: An approach to Testing Based on Combinatorial Design.*** IEEE Transactions on Software Engineering, 23 (1997), 437-444.

12. Colbourn C. J. and Dinitz J. H., ***The CRC Handbook of Combinatorial Designs***. CRC Press 1996.

13. Dill D., ***Murφ Description Language and Verifier***. http://sprout.stanford.edu/dill/murphi.html .

14. Edelman A., *The mathematics of the Pentium division bug*. SIAM Review 39 (1997) 54-67

15. Erdös P., Ko C., and Rado R., ***Intersection theorems for systems of finite sets***. Quart. J. Math. Oxford 12 (1961), 313-318.

16. Farchi E., Hartman A., and Pinter S. S., ***Using a model-based test generator to test for standards conformance***. IBM Systems Journal 41 (2002) 89-110.

17. Fisher R. A., ***An examination of the different possible solutions of a problem in incomplete blocks.*** Ann. Eugenics 10 (1940) 52-75.

18. Gal S., ***Rendezvous search on a line.*** Operations Research 47 (1999), 974-976.

19. Gargano L., Körner J., and Vaccaro U. , ***Capacities: from information theory to extremal set theory.*** J. Combinatorial Theory Series A. (to appear).

20. Gargano L., Körner J., and Vaccaro U., ***Sperner capacities***. Graphs and Combinatorics, 9 (1993) 31-46.

21. Godbole A. P., Skipper D. E., and Sunley R. A.***, t-Covering arrays: Upper bounds and Poisson approximations***. Combinatorics, Probability, and Computing 5 (1996) 105-117.

22. Greene C., ***Sperner families and partitions of a partially ordered set***. In Combinatorics (ed. M. Hall Jr. and J. van Lint) Dordrecht, Holland, 1975, 277-290.

23. Harborth H. and Kemnitz A., ***Calculations for Bertrand's Postulate***. Math. Magazine 54 (1981), 33-34.

24. Katona G. O. H., ***Two applications (for search theory and truth functions) of Sperner type theorems***. Periodica Math. Hung. 3 (1973), 19-26.

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

26. Lei Y. and Tai K. C., ***In-parameter order: A test generation strategy for pairwise testing***. in Proc. 3rd IEEE High Assurance Systems Engineering Symposium, (1998) 254-161.

27. Leveson N. and Turner C. S. *An investigation of the Therac-25 accidents*. IEEE Computer, 26 (1993), 18-41.

28. Lim W. S. and Alpern S., ***Minimax rendezvous on the line***. SIAM J. Control and Optim. 34 (1996), 1650-1665.

29. Lions J. L., *Ariane 5 Flight 501 failure, Report by the Inquiry Board.* http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html.

30. Metsch K., ***Improvement of Bruck's completion theorem***. Designs, Codes and Cryptography 1 (1991), 99-116.

31. Naor J. and Naor M., *Small-bias probability spaces: efficient constructions and applications*. SIAM J. Computing, 22 (1993) 838-856.

32. Naor M., Schulman L. J., and Srinvasan A., *Splitters and near-optimal randomization*. Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS), 1996, pp. 182-191

33. Nurmela K., ***Upper bounds for covering arrays by tabu search***. submitted.

34. Peled D. A., *Software reliability methods*. Springer, New York 2001.

35. Rényi A., ***Foundations of probability***. Wiley, New York, 1971.

36. Roux G., ***k-propriétés dans les tableaux de n colonnes; cas particulier de la k-surjectivité et de la k-permutativité***. Ph. D. Dissertation, University of Paris 6, 1987.

37. Seroussi G. and Bshouty N. H., ***Vector sets for exhaustive testing of logic circuits***. IEEE Trans. Information Theory, 34 (1988) 513-522.

38. Sloane N. J. A., ***Covering arrays and intersecting codes***. J. Combinatorial Designs 1 (1993), 51-63.

39. Stevens B., Ling A., and Mendelsohn E., *A direct construction of transversal covers using group divisible designs*. Ars Combin. (to appear).

40. Stevens B., Moura L., and Mendelsohn E., ***Lower bounds for transversal covers***. Designs, Codes, and Cryptography, 15 (1998) 279-299.

41. Tang D. T. and Chen C. L., ***Iterative exhaustive pattern generation for logic testing***. IBM J. Res. Develop. 28 (1984), 212-219.

42. Tang D. T. and Woo L. S., ***Exhaustive test pattern generation with constant weight vectors***. IEEE Trans. Computers 32 (1983) 1145-1150.

43. Web W. W., *The Vandermonde determinant*, http://www.webeq.com/WebEQ/2.3/docs/tour/determinant.html

44. West D. B.***, Introduction to Graph Theory***. Prentice Hall NJ, 1996.

45. Williams A. W.***, Determination of Test Configurations for Pair-Wise Interaction Coverage***. in Proceedings of the 13th International Conference on the Testing of Communicating Systems (TestCom 2000), Ottawa Canada, 2000, 59-74.