Locating and Detecting Arrays for Interaction Faults

Charles J. Colbourn and Daniel W. McClary
Computer Science and Engineering
Arizona State University
PO Box 878809
Tempe, AZ 85287-8809
U.S.A.

Abstract

The identification of interaction faults in component-based systems has focussed on indicating the presence of faults, rather than their location and magnitude. While this is a valuable step in screening a system for interaction faults prior to its release, it provides little information to assist in the correction of such faults. Consequently tests to reveal the location of interaction faults are of interest. The problem of nonadaptive location of interaction faults is formalized under the hypothesis that the system contains (at most) some number d of faults, each involving (at most) some number t of interacting factors. Restrictions on the number and size of the putative faults lead to numerous variants of the basic problem. The relationships between this class of problems and interaction testing using covering arrays to indicate the presence of faults, designed experiments to measure and model faults, and combinatorial group testing to locate faults in a more general testing scenario, are all examined. While each has some definite similarities with the fault location problems for component-based systems, each has some striking differences as well. In this paper, we formulate the combinatorial problems for locating and detecting arrays to undertake interaction fault location. Necessary conditions for existence are established, and using a close connection to covering arrays, asymptotic bounds on the size of minimal locating and detecting arrays are established.

1 Introduction

This is a preprint version of a paper appearing in J Comb Optim (2008) 15: 17–48.

Testing component-based systems is a challenging problem. Consider, for example, the manufacture and delivery of a home computing system. Many hardware components are intended to be interoperable, so a variety of choices may be available for the processor, primary memory, cache memory, disk interface, CD/DVD device, and the like. Once a hardware platform is selected, standard software tools also exhibit a wide variety of choices for each component, such as operating system, web browser, email system, and the like. Once software is installed, the system itself will then function in an environment controlled by many factors, each with multiple options – for example, the system may function as a server or a client in a networked environment.

The customer, quite reasonably, expects the system to function correctly on delivery. Yet in a system of even moderate complexity in terms of the options available, the manufacturer cannot be expected to have experience with all of the possible systems that could be built. There are just too many of them. Naturally the manufacturer is expected provide only those options that have survived rigorous unit testing, so that the option works according to specification in isolation. Moreover, for components in which there is a documented interaction (such as the interface between cache and main memory), one can expect that the manufacturer conducts thorough integration testing to ensure that these two components interact as expected.

Unanticipated interactions that disrupt proper system operation pose the real problems here. While fault models and user models can suggest areas for further testing, they focus on interactions that the models predict, and hence can miss important interactions. To give a simple example, if an operating system requires certain data to be in cache, and a disk interface reserves cache for direct memory access, but the cache is too small to support both, this may cause total system failure or degradation of system performance. Attributing the fault to any one of the cache, operating system, or disk interface alone does not adequately explain the fault.

One reason why user modelling and fault modelling are prevalent is that it is not possible, in general, to examine every possible interaction without conducting exhaustive testing; then treating those that appear to be the most likely is natural. Among those that our models do *not* anticipate, which interactions are most likely to cause faults? A single system may involve millions of interactions of hundreds of components, yet the explanation for the failure of such a system is typically (not always) due to the faulty interaction of a small number of them. Interaction testing proceeds by setting a limit on the number of components involved in interactions to be examined, and ensuring that testing includes systems with all possible interactions of a number of components up to this specified size.

Let us introduce this testing problem more formally. There are k factors $F_1, \ldots, F_k$. Each factor F_i has a set $S_i = \{v_{i1}, \ldots, v_{is_i}\}$ of s_i possible values (*levels*). A *test* is an assignment, for each $i = 1, \ldots, k$, of a level from $v_{i1}, \ldots, v_{is_i}$ to F_i. A test, when executed, can *pass* or *fail*. For any subset $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, k\}$ and levels $\sigma_{i_j} \in S_{i_j}$, the set $\{(i_j, \sigma_{i_j}) : 1 \leq j \leq t\}$ is a *t-way interaction*, or an interaction of *strength t*. Thus a test on k factors contains (*covers*) $\binom{k}{t}$ interactions of strength t. A *test suite* is a collection of tests; the *outcomes* are the corresponding set of pass/fail results. A fault is evidenced by a failure outcome for a test; however the fault is rarely due to a complete k-way interaction; rather it is the result of one or more faulty interactions of strength smaller than k covered in the test. All tests chosen are executed in parallel, so that testing is *nonadaptive*. Unlike *adaptive* testing in which tests are chosen based on the outcomes of the executions of prior tests, here no test outcomes are available to guide the section of tests.

Given a test suite, and the set of interactions causing faults, the outcomes can be easily calculated: A test fails exactly when it contains at least one of the faulty interactions, and does not fail otherwise. In order to observe an interaction fault, it is necessary that the interaction be covered by at least one test in the test suite. With no restriction on the interactions that can cause faults, the best one can do is to form all $\prod_{i=1}^{k} s_i$ possible tests, the *exhaustive* test suite. Rarely, however, do faults arise from interactions of "high" strength, and therefore an upper bound t is placed on the strength of interactions to be covered.

We adopt a matrix formulation. An array with N rows, k columns, and symbols in the ith column chosen from an alphabet S_i of size s_i is denoted as an $N \times k$ array of type $(s_1, \ldots, s_k)$. A *t-way interaction* in this array is a choice of t columns $i_1, \ldots, i_t$, and the selection of a level $\sigma_{i_j} \in S_{i_j}$ for $1 \leq j \leq t$, represented as $T = \{(i_j, \sigma_{i_j}) : 1 \leq j \leq t\}$. For such an array $A = (a_{xy})$ and interaction T, define $\rho(A, T) = \{r : a_{ri_j} = \sigma_{i_j}, 1 \leq j \leq t\}$, the set of rows of A in which the interaction is covered. For a set of interactions $\mathcal{T}$, $\rho(A, \mathcal{T}) = \bigcup_{T \in \mathcal{T}} \rho(A, T)$. Then locating faults requires that $\mathcal{T}$ be recovered from $\rho(A, \mathcal{T})$. Arrays for this purpose are defined in §2.

The arrays to be defined relate to numerous other combinatorial structures, to which connections are explored in the subsequent sections. The outcomes of a test suite execution must reveal the presence of faulty interactions among those considered. Hence every possible interaction must appear in at least one test. An $N \times k$ array A of type $(s_1, \ldots, s_k)$ is a *mixed level covering array* of strength t, denoted $\mathrm{MCA}(N; t, (s_1, \ldots, s_k)$, if for every t-way interaction T, one finds that $|\rho(A, T)| \geq 1$, i.e. every t-way interaction appears at least once. These have been very extensively studied; a brief survey is given in §3. The notation $\mathrm{CA}(N; t, k, v)$ is used when all factors have the same number of levels, i.e. when $s_1 = \cdots = s_k = v$.

If faults arise from one or more interactions of strength at most t, the outcomes from the tests in a covering array will indicate the presence of faults. However the *location* and *magnitude* of the interactions causing the faults may be far from clear. In a covering array, it is a frequent occurrence that t-way interactions are covered only once; yet faults from two interactions that each appear in the same test, and only in that test, cannot be distinguished. This measurement of interactions is the subject of an extensive field, the statistical design of experiments. The techniques developed there are targeted towards outcomes of a continuous nature, such as execution time, or amount of memory consumed, rather than binary 'pass/fail' outcomes. In §4 techniques are examined from the design of experiments that facilitate measurement – and hence location – of interactions, rather than simply indicating the presence of one or more faulty interactions.

On the one hand, covering arrays indicate the presence of faulty interactions, while designed experiments aim to quantify their contribution to the outcomes. In our testing environment, no such quantification is reasonable, since an interaction is faulty or not, and a test passes or fails. Viewed as a problem of locating defectives in a population using tests that "pool" items for testing establishes a further connection, this time with combinatorial group testing. This connection is explored in §5.

Having developed this extensive set of relationships with other combinatorial structures, in §6 we provide a collection of illustrative examples and in §7 we determine when locating arrays can exist for a broad class of sets of interaction faults. Finally in §8 we establish that for fixed number and strength of interactions, the number of tests in a locating or detecting array grows logarithmically in the number of factors.

2 Definitions: Locating and Detecting Arrays

Let $\mathcal{I}_t$ be the set of *all* t-way interactions for an array of type $(s_1, \ldots, s_k)$, and let $\overline{\mathcal{I}_t}$ be the set of all interactions of strength *at most* t. Consider an interaction $T \in \overline{\mathcal{I}_t}$ of strength less than t. Any interaction T' of strength t that contains T necessarily has $\rho(A, T') \subseteq \rho(A, T)$, and hence when T is faulty we are unable to determine whether or not T' is also faulty; call a subset $\mathcal{T}'$ of interactions in $\mathcal{I}_t$ *independent* if there do not exist $T, T' \in \mathcal{T}'$ with $T \subset T'$. In general, some interactions in $\mathcal{I}_t$ (or perhaps $\overline{\mathcal{I}_t}$) are believed to be faulty, but their number and identity are unknown. As discussed later, if there is no information on the number of such faulty interactions, it is impossible to identify them precisely from the outcomes. We therefore suppose that *a priori* information on the number is available; a number d of faulty interactions is specified, and when d is an upper bound on such number we employ the notation $\overline{d}$. Of course it is often unreasonable to expect that such *a priori* information on the strengths and number of faults is available; we return to this point later.

Write $\rho(A, \mathcal{T}') = \bigcup_{T \in \mathcal{T}'} \rho(A, T)$. An array A is (d, t)-*locating* if

$$\rho(A, \mathcal{T}_1) = \rho(A, \mathcal{T}_2) \Leftrightarrow \mathcal{T}_1 = \mathcal{T}_2$$

whenever $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{I}_t$, $|\mathcal{T}_1| = d$, and $|\mathcal{T}_2| = d$. The notation is extended to permit interactions of strength at most t by writing $\hat{t}$ in place of t, and permitting instead that $\mathcal{T}_1, \mathcal{T}_2 \subseteq \overline{\mathcal{I}_t}$. When in addition $\mathcal{T}_1$ and $\mathcal{T}_2$ are required to be independent, the notation $\overline{t}$ is used instead. In a similar manner, the definition is extended to permit sets of at most d interactions by writing $\overline{d}$ in place of d, and permitting instead that $|\mathcal{T}_1| \leq d$ and $|\mathcal{T}_2| \leq d$.

Using a (d, t)-locating array A to conduct tests, if there is any set of d interactions of strength t that produce exactly the outcomes obtained, then there is exactly one such set of interactions. Hence in principle the location of the interactions causing the faults can be calculated from the outcomes.

The *a priori* assumptions on number and strength of interactions poses a serious practical concern in testing. How could one recover if these bounds are supplied incorrectly? In essence, one wants a guarantee that if more than d interactions are present, or one has strength greater than t, this can be distinguished from an explanation of the outcomes involving at most d interactions each of strength at most t. As we have seen (implicitly), any outcome involving an interaction T in its explanation will be equally well explained by adjoining any further interaction T' having $T \subset T'$ to the explanation, and hence interactions of higher strength can easily be masked by those of lower strength.

In essence, locating faults that are necessarily hidden by faults of smaller strength contained within the fault cannot be done, and so when faulty interactions are not independent, it is required to know the strengths of the faults. One might hope that requiring the faults found to be an independent set resolves this problem. But unless an exhaustive array is employed, some interaction does not appear in any test – and hence if faulty cannot be located. For this reason, while it may happen that the presence of interactions of strength greater than t will be indicated by the outcomes, this cannot be ensured.

On the other hand, determining whether there are more than d t-way interactions is an easy extension. For if $\mathcal{T}$ is a set of at most d t-way interactions and $\mathcal{R}$ is a set of more than d t-way interactions, every $R \in \mathcal{R}$ satisfies $\rho(A, R) \subseteq \rho(A, \mathcal{T})$. In analogy with specializing union-free families to cover-free families (discussed in §5), we introduce a further set of definitions. An array A is (d, t)-*detecting* if

$$\rho(A, T) \subseteq \rho(A, \mathcal{T}_1) \Leftrightarrow T \in \mathcal{T}_1$$

whenever $\mathcal{T}_1 \subseteq \mathcal{I}_t$, $|\mathcal{T}_1| = d$, and $T \in \mathcal{I}_t \setminus \mathcal{T}_1$. The notation is extended to permit interactions of strength at most t by writing $\hat{t}$ in place of t, and permitting instead that $\mathcal{T}_1 \subseteq \overline{\mathcal{I}}_t$ and $T \in \overline{\mathcal{I}}_t$. When in addition $\mathcal{T}_1 \cup \{T\}$ is required to be independent, the notation $\overline{t}$ is used instead. Similarly the notation $\overline{d}$ is used to represent at most d interactions.

While detecting arrays may necessitate more rows in order to locate, the algorithm for location is straightforward. One simply considers all of the outcomes. When a test passes, all interactions in it are known not to be faulty. Those interactions not found to pass in this way form a set of *candidate faults*. If there are at most d candidate faults, then a d-detecting array identifies the candidate faults as the set of faulty interactions. If, on the other hand, there are more than d, there is no set of d of fewer faults explaining the outcomes; but we cannot be sure that all candidate faults are in fact faulty (nevertheless, all faulty interactions must appear among the candidate faults).

The latter observation is crucial. If a set of d or fewer faults is found, then no other fault can be present. In this way, the *a priori* limit on d can be validated. Indeed this serves as an indicator of the relationship to combinatorial group testing for defective items, where the corresponding observation arises as well [24, p. 134].

001221	010122
021012	022101
012210	102112
120211	112021
111202	121120
112002	121200
102120	100212
120021	010220
001022	020102
022010	002201

Shown on the left is an example array for a system with six factors. The first has two levels (0 and 1), while the remaining factors have three levels each (0, 1, and 2). Each of the twenty rows displayed is a test. This array is (1,1)-locating because all of the 1-way interactions appear in a different set of tests; indeed it is (1,1)-detecting, because the set of tests involving any one 1-way interaction never contains the set of tests involving another. It is (2,1)-locating because the union of the tests containing either or both of two specified 1-way interactions never equals the union for another. However it is not (2,2)-detecting because the two 1-way interactions (1,0) and (1,1) together appear in all tests, and hence include the tests in which any other 1-way interaction appears.
The array is (1,2)-locating as well, but not (1,2)-detecting. These statements can be verified by determining the sets of tests in which each of the 120 2-way interactions arise, noting that no two are equal but proper containment does occur.

3 Covering Arrays

Covering arrays have an extensive history, and we do not attempt a survey here. They have arisen in numerous combinatorial problems, but our primary concern is with fault testing. The goal of testing with covering arrays, or interaction test suites, is to *indicate the presence or absence of faults*, not their location, and not their magnitude. Fault location evidently requires that the presence of faults be determined, and hence covering arrays provide a minimal type of fault location. Indeed when the outcomes of the tests are known, an adaptive strategy can test further by restricting attention to only those interactions appearing in faulty tests. Nonadaptive strategies, on the other hand, necessitate further tests in the array itself. Nevertheless, since locating and detecting arrays are also covering arrays, we review the current state of affairs on covering arrays to provide context for fault location.

The combinatorics of covering arrays is treated in [18, 34], although no textbook treatment of the subject is available; we just summarize the principal research directions. An early investigation of covering arrays appears implicitly in Marczewski [51]. Rényi [60] determined sizes of covering arrays for the case $t = v = 2$ when N is even. Kleitman and Spencer [43] and Katona [40] independently determined covering array numbers for all N when $t = v = 2$. They determined the exact relationship between N and k: $k = \binom{N-1}{\lceil \frac{N}{2} \rceil}$.

For large k, N grows logarithmically. The construction is straightforward. Form a matrix in which the columns consist of all distinct binary N-tuples of weight $\lceil \frac{N}{2} \rceil$ that have a 0 in the first position. In 1990 Gargano, Körner and Vaccaro [30] gave a probabilistic bound when $t = 2$ and $v > 2$:

$$N = (\frac{v}{2} \log k)(1 + o(1))$$

Now we explore a dual formulation. Let C be an $N \times k$ covering array. Suppose that rows are indexed by a set R of size N. Then each column can be viewed as a partition of R into exactly v classes $(M_1, \ldots, M_v)$; the class M_i of a symbol $r \in R$ is determined by the level i appearing in row r in the chosen column. In this manner, such an array gives a collection $\mathcal{P} = \{R_1, R_2, \ldots, R_k\}$ of partitions of R. A family of partitions is *t-qualitatively independent* when for every t of the partitions $R_{i_1}, \ldots R_{i_t}$, and for every choice of classes $M_{i_j} \in R_{i_j}$, for $1 \leq j \leq t$, we find that $\bigcap_{j=1}^{t} M_{i_j} \neq \emptyset$. It follows that covering arrays of strength t having N rows are the same as t-qualitatively independent partitions of a set of size N. Many of the early results were established using this vernacular, but we for the most part translate to the language of covering arrays. Poljak, Pultr, and Rödl [58] and Körner and Lucertini [44] discuss combinatorial problems related to qualitative independence.

Covering arrays appear in other mathematical disguises as well. A *(k,t)-universal set* is a subset of $\{0,1\}^k$ such that the projection on every t coordinates contains all 2^t combinations. Hence it is a CA$(N; t, k, 2)$. Naor and Naor [55] establish that (k,t)-universal sets arise as probability spaces with limited independence; indeed these have been extensively studied as ϵ-biased arrays [1, 47, 55]. Bierbrauer and Schellwatt [3] extend this framework to more than two levels per symbol.

The nomenclature *t-surjective array* for a covering array of strength t is also used, to indicate that on each t columns, every possible outcome arises. See [11, 14, 28, 64], for example.

3.1 The Basics

Let $\mathsf{CAN}(t,k,v)$ be the minimum number N of rows in a CA$(N; t, k, v)$. An obvious lower bound is $v^t \leq \mathsf{CAN}(t,k,v)$. More generally for an MCA$(N; t, k, (v_1, v_2, \ldots, v_k))$, $\prod_{i=1}^{t} v_i$ is a lower bound on the size of the mixed covering array. Although the $\{v_i\}$ are listed in a specified order, reordering the $\{v_i\}$ in the definition results in a simple column reordering of the MCA. For this reason, we present these sizes in any convenient order. In this way, the stated bound can be treated as the product of the t largest values among the $\{v_i\}$.

An equally obvious lower bound results from the requirement that every two columns be distinct; indeed columns with v_i and v_j symbols, $v_i \leq v_j$, differ in at least $v_i(v_j - 1)$ rows; hence whenever $t \geq 2$ and all numbers of levels are required to be at least 2, $N \geq \log k$.

Evidently if any of the $\{v_i\}$ is 1, it can be omitted provided that the number of factors is at least two. With this in mind, a simple inequality establishes that

$$\mathsf{CAN}(t,k,(v_1, v_2, \ldots, v_k)) \leq \mathsf{CAN}(t,k,(v_1+1, v_2, \ldots, v_k)) \tag{1}$$

and consequently

$$\mathsf{CAN}(t,k-1,(v_2, \ldots, v_k)) \leq \mathsf{CAN}(t,k,(v_1, v_2, \ldots, v_k)) \tag{2}$$

Suppose A is a covering array CA$(N; t, k, v)$ and let i be any row and x any symbol. Then the $(k-1) \times N'$ subarray obtained by deleting row i from A and keeping only those columns of A that had symbol x in row i is a CA$(N'; t-1, k-1, v)$, where N' is the number of occurrences of x in row i. Hence

$$\mathsf{CAN}(t-1,k-1,v) \leq \frac{1}{v}\mathsf{CAN}(t,k,v). \tag{3}$$

By the same token,

$$\mathsf{CAN}(t, k, (v_1, v_2, \ldots, v_k)) \quad \geq \quad v_1 \cdot \mathsf{CAN}(t-1, k-1, (v_2, \ldots, v_k)) \tag{4}$$

More sophisticated bounds have been the subject of much research. Godbole, Skipper, and Sunley [33] examine the random process of choosing, in each entry of an $N \times k$ array, each of v possible levels equiprobably. They establish that when N is large enough with respect to t, k, and v, such a random array has nonzero probability of being a $\mathrm{CA}(N; t, k, v)$. It follows from their results that

$$\mathsf{CAN}(t, k, v) \leq \frac{(t-1)\log k}{\log \frac{v^t}{v^t - 1}}(1 + \mathrm{o}(1)).$$

In [2], it is established that

$$\mathsf{CAN}(t, k, 2) \leq 2^t t^{\mathrm{O}(\log t)} \log k.$$

Gargano, Körner, and Vaccaro [29] show that the ratio of $\mathsf{CAN}(2, k, v)$ to $\log k$ is asymptotic to $\frac{1}{2}v$. Indeed a stronger version in [30] establishes the same result when only a small fraction of the pairs need to be covered. For higher strength, considering the largest k for which a $\mathrm{CA}(N; t, k, v)$ exists, it has been established that k is at least $\frac{et}{v}e^{\frac{N}{tv^t}}$ and at most $\frac{\kappa_{v,N}}{\sqrt{N}}4^{\frac{N}{v^{t-1}}}$, where $\kappa_{v,N}$ is a constant depending only upon v and N; see [58, 59].

Lower bounds are in general not well explored.

3.2 Combinatorial Constructions

An *orthogonal array* $OA_\lambda(N; t, k, v)$ is an $N \times k$ array. In every $N \times t$ subarray, each t-tuple occurs exactly λ times, where $\lambda = \frac{N}{v^t}$. The parameter t is the *strength*; k is the number of *factors*; and v is the number of levels associated with each factor, the *order*. Evidently an orthogonal array $\mathrm{OA}(N; t, k, v)$ is a covering array, and when $N = v^t$ it is optimal. The theory of orthogonal arrays is well developed; indeed a recent book treats many facets of their construction and use [36]. Although orthogonal arrays both provide many of the best constructions, and motivate many of the published methods, we do not delve into them here but instead refer the reader to [34] for an overview in the context of covering arrays, to [20] for $\mathrm{OA}(2, k, v)$s, and to [36] for higher strength.

Direct constructions using group divisible designs, orthogonal arrays, and projective and affine geometries are described in [18, 34]. These provide relatively few examples of the smallest covering arrays—but the ones provided are often optimal or close to optimal.

Following Roux [61, 67], many "cut-and-paste" recursive constructions have been developed. For $t = 2$ the best available Roux-type constructions appear in [21]; for $t = 3$ in [22]; for $t = 4$ in [22, 53], and for $t \geq 5$ in [52, 53]. The basic strategy in each is to form a number ℓ of copies of a t-covering array on k factors to form an array on $k\ell$ factors, simply juxtaposing the copies. While such juxtaposition covers many of the t-tuples, typically some remain to be covered by covering arrays of lower strength. The refinements in this method have resulted primarily from the careful analysis of the extra rows needed.

A different inflation strategy is very effective, especially for large covering arrays. Two examples follow.

The *Turán number* $T(t,n)$ is the largest number of edges in a t-vertex simple graph having no $(n+1)$-clique. Turán showed that a graph with the $T(t,n)$ edges is constructed by setting $a = \lfloor t/n \rfloor$ and $b = t - na$, and forming a complete multipartite graph with b classes of size $a+1$ and $n-b$ classes of size a (see [34]).

Theorem 3.1. *[34] If a $CA(N;t,k,v)$ and a $CA(k^2;2,T(t,v)+1,k)$ both exist, then a $CA(N \cdot (T(t,v)+1);t,k^2,v)$ exists.*

Perfect hash families are well studied combinatorial objects. A *t-perfect hash family $\mathcal{H}$*, denoted PHF$(n;k,q,t)$, is a family of n functions $h : A \mapsto B$, where $k = |A| \geq |B| = q$, such that for any subset $X \subseteq A$ with $|X| = t$, there is at least one function $h \in \mathcal{H}$ that is injective on X. Thus a PHF$(n;k,q,t)$ can be viewed as an $n \times k$-array $\mathcal{H}$ with entries from a set of q symbols such that for any set of t columns there is at least one row having distinct entries in this set of columns.

Theorem 3.2. *(see [3, 53]) If a PHF$(s;k,m,t)$ and a $CA(N;t,m,v)$ both exist then a $CA(sN;t,k,v)$ exists.*

For the current status of existence of perfect hash families see [76].

3.3 Algorithmic Methods

Algorithms and tools to construct interaction test suites are of substantial interest. Available algorithms arise from many different sources. TConfig [78] develops a recursive construction method based on orthogonal arrays. Exact algorithms (integer programming, backtracking) have met success only for very small problems [18]. Constraint programming has been applied for strengths greater than two [37]. Meta-heuristic search has been quite effective. Simulated annealing (SA) [16, 70] and tabu search [56] have produced many of the smallest available test suites. In these approaches, transformations are repeatedly applied to the current putative test suite, and a transformation is “accepted” using an heuristic decision rule (see [15, 16, 56]).

In simulated annealing [16], a test suite goes through a series of transformations. A typical transformation is to change a single level in one test. The current cost is measured as the number of t-way interactions not covered. When a transformation reduces this cost, it is performed. Otherwise, it is performed with a probability that changes dynamically according to a given “cooling schedule”. In cite [17], augmented annealing combines simulated annealing with combinatorial constructions to improve both accuracy and efficiency.

In tabu search [56], transformations are also applied to a putative test suite. Cost is again the number of uncovered t-way interactions. The least cost move is selected from among those that are not *tabu*. Typically a tabu move is one that would return to a test suite that is included among those seen within the last T moves; T is the length of history maintained.

A greedy algorithm with both horizontal and vertical growth is IPO [72]. However, most greedy algorithms in the literature use a one-test-at-a-time strategy. These are exemplified by AETG [13], TCG [75], and DDA [8].

If the size of the test suite is the overriding concern, meta-heuristic search often yields the best results. The substantial time required (and perhaps the complexity of implementing search methods) has led to the widespread use of simpler heuristics, such as hill-climbing [16] and greedy methods. See [16, 17, 56] for data on accuracy and execution time of simulated annealing and tabu search.

3.4 Applications

The concept of pairwise coverage has been used across many disciplines including medicine, agriculture and manufacturing [36]. It has entered the software testing community, appearing in practitioner's guidebooks and provided in simple spreadsheets. The use of covering arrays in software testing was pioneered by Mandl [50] and Brownlie et al. [7, 57]. Empirical results indicate that testing of all pairwise interactions in a software system indeed finds a large percentage of existing faults [23, 45, 46]. Indeed, Burr et al. [9] provide more empirical results to show that this type of test coverage leads to useful *code* coverage as well. Dalal *et al.* present empirical results to argue that the testing of all pairwise interactions in a software system finds a large percentage of the existing faults [23]. Dunietz *et al.* link the effectiveness of these methods to software code coverage. They show that high code block coverage is obtained when testing all two-way interactions, but higher subset sizes are needed for good path coverage [25]. Kuhn *et al.* examined fault reports for three software systems. They show that 70% of faults can be discovered by testing all two-way interactions, while 90% can be detected by testing all three way interactions. Six-way coverage was required in these systems to detect 100% of the faults reported [45]. This study was followed by similar experiments, such as one of 109 software-controlled medical devices that were recalled by the U.S. Food and Drug Administration (FDA) [46]. These experiments found that 97% of the flaws in these 109 cases could be detected with pair-wise testing of parameter settings. Only three devices required coverage higher than two.

Williams et al. [78] quantify the coverage for a particular interaction strength. For instance, if we have four factors, any new test case can contribute at most $\binom{4}{2}$, or 6 new covered pairs. Further, if each factor has three levels, there are a total of $\binom{4}{2}\ 3^2 = 54$ possible pairs that must be covered. Therefore any one new test case increases our coverage by at most 11.1% [78]. A similar method is described by Dunietz et al. [25]. We would expect that a real testing environment has the ability to capture variable coverage requirements for a given test suite.

3.4.1 Hardware Testing

Imagine a circuit with k inputs that we are to test. Within the circuit, the input signals interact through arithmetic and logical operations to determine an output vector. From the specification, we know the output vector that should be produced, but errors may occur. As with software, we expect errors to be evinced by setting a fraction of the inputs to specified high or low values. Tang *et al.* [73] study hardware (circuit testing) in this environment, proposing test coverage that includes each of the 2^t input settings for each subset of t inputs. Seroussi and Bshouti [64] give a comprehensive treatment. In each case, the test suite is a binary covering array of strength t.

3.4.2 Testing Advanced Materials

Cawse [10] reports on experimental plans for mixture experiments, in which materials are combined to yield improved strength, flexibility, melting point, and the like. In this application, *avoiding* certain combinations is not just desirable – rather it is necessary, as certain mixtures can be explosive, or toxic. With this in mind, the goal is to consider a representative sample of mixtures, under a variety of controlled environmental conditions. Cawse advocates the use of covering arrays for such experimentation.

3.4.3 Interactions Regulating Gene Expression

Regulation of developmental and biological processes depends upon certain signals (hormones) impact the expression of a particular gene. Multiple signals may interact in regulation, by jointly inhibiting or enhancing gene expression. Numerous examples are given in [65]. Identification of signal interactions cannot be achieved by examining all possible signal combinations, but a (by now) familiar theme emerges. Interactions among few signals are those of most interest. Covering arrays provide precisely the experimental plan to ensure that all "small" potential interactions are explored. Shasha *et al.* [65] provide details on the application.

3.5 Summary

Tables of the best known covering array numbers are given in [19] for $t = 2$, [22] for $t \in \{3, 4\}$, and [77] for $t = 5$. Except when $t = k = 2$, the size of a minimum covering array remains open, although when t is small the existing techniques appear to be quite effective.

In the context of locating arrays, however, this extensive literature on covering arrays leaves many questions unanswered. Most particularly, the effectiveness of covering arrays at locating faults rather than indicating their presence has not been studied from a combinatorial viewpoint; indeed even their ability to narrow the possible faults to a "small" set does not appear to have been studied. However it has been shown that covering arrays are competitive with D-optimal designs from the design of experiments when used to measure interactions [39]. In fact the most suitable covering arrays are experimentally observed to be those of strength t that evenly balance the occurrences of $(t + 1)$-tuples [38], a feature that has been used in heuristic algorithms for the construction of covering arrays [35]. Later we see that this condition arises naturally when location of faults is required. The literature on covering arrays therefore provides a guide of sorts, suggesting possible extensions from covering to locating arrays.

4 Designed Experiments

If fault testing via minimum covering arrays lie at one end of the testing spectrum, statistical *design of experiments* (DOE) lies at the opposite. While DOE typically employs test suites that are substantially larger than those used in fault testing, the aim is very different. DOE aims to provide test suites such that the system in question can be (approximately) modeled.

DOE focuses on designing test suites that allow *analysis of variance* (ANOVA) testing, least-squares estimation, and other methods to produce statistically significant test results and regression models in a minimum number of tests. The core of most designed experiments is the *full factorial* design, written v^k, which is equivalent to the exhaustive test suite E for factors $F_1 \ldots F_k$ with $v = s_1 = s_2 \cdots = s_k$. Often all factors are restricted to 2 or 3 symbols ($\{-1, +1\}$ and $\{-1, 0, +1\}$), leading to the more common notation as 2^k and 3^k factorial designs.

To probe the whole of the designed experiments literature is well beyond the scope of this work. For a survey, see [54]. Traditionally, test cases are added to or subtracted from a full factorial design to achieve certain properties in the subsequent statistical analysis. *Center* and *axial* runs as well as replicated suites are used to detect potential curvature in the response of the system and lower prediction variance for any regression models fitted. Subtraction of runs, on the other hand, is used to minimize the number tests when cost prohibits use of exhaustive suites.

In order to maintain the necessary properties for statistical analysis, runs are usually reduced in groups and along guidelines. These reductions result in test suites in which k factors are tested in what would be an exhaustive suite for $k - p$ factors. In the DOE literature, designs are considered $\frac{1}{v^p}$ fractions of the v^k design. Typically, these designs are limited to $v = 2$ and referred to as 2^{k-p}. Naturally, the saturation of a smaller design with a large number of factors causes a certain amount of information to be obscured. For the case of these *fractional factorial designs*, the effects of certain factors and their interactions are confounded with one another. In the design of experiments literature this is called effect *aliasing.*

Regardless of the reduction strategy employed, designs must maintain full rank in order to solve for the unknowns representing the measurement of each interaction. Intuitively, consider a t-way interaction T, and suppose that f is a factor with s values that is not represented in T. Let S be the set of s values for factor f. By definition, the contribution of interaction T is equal to the sum of the contributions for the s interactions $\{T \cup \{\sigma\} : \sigma \in S\}$. Hence knowing the contribution for T and the contributions for $s-1$ $(t+1)$-way interactions that extend it to factor f, we can compute the last. Thus the number of independent variables (and hence the rank, and hence the number of tests or runs required) can be calculated as

$$\sum_{i=0}^{t} \sum_{F \subseteq \{1,\ldots,k\}, |F|=i} \prod_{f \in F} (s_f - 1), \tag{5}$$

where the empty product is as usual taken to be 1. Indeed distinguishing among interactions for k factors (having at least two, but at most some fixed number of, levels) requires a number of runs that is $\Theta(k^t)$, and hence experimental designs are usually restricted to few factors and small interactions.

4.1 Linear Contrasts and Factor Effects

Many techniques exist for computing the effect of the factors in a designed experiment on their response, among them ANOVA and least-squares estimation. However, the most basic means of determining factor effects is via *linear contrasts*.

Generally speaking, a contrast Γ is a linear combination of terms as in equation (6), with parameter value $\sigma_{i_j} \in S_i$, the result for the ith test case μ_i and $\sum_i^k \sigma_{i_j} = 0$.

$$\Gamma = \sum_i^k \sigma_{i_j} \mu_i \tag{6}$$

Contrast computation is the reason that designed experiments adopt a {-1, +1} notion instead a standard binary coding. In fact, contrasts prove valuable in vetting any number of hypotheses regarding a test suite. Contrasts are typically used for testing the effect of a factors or t-way interactions that have been previously identified as having some significance. However, Scheffé's method [63] allows comparisons of all contrasts present in a test suite. Thus, even for test suites with pass-fail responses, the effects of fault-causing factors and interactions can be determined, provided the response is coded appropriately.

Speaking very generally, to determine the relative effect of a set of interactions $\mathcal{T}$ one needs only to compare the magnitudes of $\Gamma_{\mathcal{T}}$ the contrast for all $T \in \mathcal{T}$ with $\Gamma_{\mathcal{R}}$, where $\mathcal{R}$ is the set of all t-way interactions not in $\mathcal{T}$. Similarly, to determine the effect of a single interaction $T \in \mathcal{T}$, construct the set of interactions $\mathcal{R}$ such that $\{T\} = \mathcal{T}/\mathcal{R}$ and compare the magnitudes of $\Gamma_{\mathcal{T}}$ and $\Gamma_{\mathcal{R}}$.

4.2 Effect Aliasing

Subtracting test cases from full factorial designs is common practice when cost must be minimized and a limited number of interactions require examination. However, use of fractional factorial and other small designs introduces effect aliasing in the design. Wise application of effect aliasing often helps minimize the number of total test cases in a suite without sacrificing its ability to estimate important factors and interactions. Of course, use of reduced designs can prevent accurate estimation of factors if employed haphazardly.

To understand how aliasing manifests itself in designs, and to determine alias structure for a given design, we next discuss aliasing and a mechanism for determining aliasing for any test suite.

Aliasing, as it is understood within DOE, is the *confounding* (i.e., combining) of a factor or interaction's estimated effect with other factors or interactions of the same or higher order. Mathematically, aliasing occurs when a linear dependence exists among two or more columns of a test suite. Naturally, such a dependency alters what can be ascertained via either linear contrasts or least-squares estimation. Reduced designs that contain no confounding among factors and interactions of interest are useful tools, and efforts to catalog such designs are extensive.

Common fractional factorial designs are cataloged by *resolution*, a notation designed to indicate the aliasing properties of the design. Designs are typically referred to as being of resolution III, IV, or V. Resolution III designs confound factor effects with 2-way interactions, and 2-way interactions with one another. Resolution IV designs confound 2-way interactions with one another, but main effects are individually estimable. Resolution V designs confound 2-way interactions with 3-way interactions.

While resolution indices are useful for choosing a reduction in runs from the exhaustive test suite, they require the designer to determine *a priori* what factors and interactions will be aliased in the design. Without preexisting information about the aliasing structure of a design, it can be difficult to grasp what confounding may exist. Moreover, for test suites with an arbitrary number of cases removed from the exhaustive suite, the resolution scheme provides no means of determining the alias relationships.

Bisgaard's work [4] provides a method for computing an *alias matrix* $\mathcal{A}$, which quantifies the confounding between the set of factors in a test suite and a set of interactions. The method is then applied to Taguchi's [71] designs to illustrate its applicability [5]. While Bisgaard's illustrates only the case in which only alias relationship with the set of 2-way interactions in a design, the technique is applicable to any set of interactions among factors. The remainder of the section details the method.

Suppose that an $N \times k$ design matrix $\mathbf{X}_1$ is considered sufficient for a test suite. Let some numbers of factors $F_1 \ldots F_i$ and interactions $I_j \ldots I_k$ be assigned to the columns of $\mathbf{X}_1$. This assignment presumes that the vector of responses $\mathbf{y}$ is a function of the form $\mathbf{y} = \mathbf{X}_1\hat{\boldsymbol{\beta}}_1 + \varepsilon$, where $\hat{\boldsymbol{\beta}}_1$ is the $1 \times k$ vector of estimated coefficients for the k factors and interactions represented in $\mathbf{X}_1$. For this model the expected value function for $\hat{\boldsymbol{\beta}}_1$ is as in equation (7). The expected value function in (7) illustrates that if the test design represented in $\mathbf{X}_1$ encompasses all of the significant factors and interactions in the system, then the coefficients estimated in $\hat{\boldsymbol{\beta}}_1$ are correct and unbiased with respect to random error.

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}_1) &= E[(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'E(\mathbf{y})] \\ &= E[(\mathbf{X}_1'\mathbf{X}_1)^{-1}(\mathbf{X}_1'\mathbf{X}_1)\boldsymbol{\beta}_1 + (\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'\varepsilon] \\ &= \boldsymbol{\beta}_1 \end{aligned} \tag{7}$$

However, assume that the system in question is affected by the factors and interactions assigned to the columns of $\mathbf{X}_1$ as well as a set of interactions not assigned to columns of $\mathbf{X}_1$. Then $\mathbf{y} = \mathbf{X}_1\boldsymbol{\beta}_1 + \mathbf{X}_2\boldsymbol{\beta}_2 + \varepsilon$. Let the values of the unassigned interactions be assigned to the columns of a second design matrix $\mathbf{X}_2$. Then, the expected value function for $\hat{\beta}_1$ becomes equation (8).

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}_1) &= E[(\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'E(\mathbf{y})] \\ &= (\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'\mathbf{X}_1\boldsymbol{\beta}_1 + (\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'\mathbf{X}_2\boldsymbol{\beta}_2 \\ &= \boldsymbol{\beta}_1 + \mathcal{A}\boldsymbol{\beta}_2 \end{aligned} \tag{8}$$

Here $\mathcal{A} = (\mathbf{X}_1'\mathbf{X}_1)^{-1}\mathbf{X}_1'\mathbf{X}_2$ is the *alias matrix*. $\mathcal{A}$ captures the aliasing between the factors and interactions in $\mathbf{X}_1$ and $\mathbf{X}_2$. Equation (9) illustrates the relationship between k factors and their 2-way interactions, but can be generalized such that $\mathbf{F}$ contains the factors and interactions tested in $\mathbf{X}_1$ and $\mathbf{F}'$ the interactions represented in $\mathbf{X}_2$.

$$\begin{aligned} \mathbf{F} \quad + \quad \mathcal{A}\mathbf{F}' &= \\ \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_k \end{pmatrix} \quad + \quad \mathcal{A} &\begin{pmatrix} F_1F_2 \\ F_1F_3 \\ \vdots \\ F_{k-1}F_k \end{pmatrix} \end{aligned} \tag{9}$$

Bisgaard's method for identifying alias relationships was developed using a standard statistical coding (i.e., {-1, 0, +1}) and multiplicative interactions. However, the relationship between equations (7) and (8) holds no matter the coding or interactions. For instance, our discussion of (d, t)-detecting/locating designs considers maximum binary interactions. Other types of interactions are both valid and equally likely to be confounded with the factors represented in the test suite.

Srivastava [69] pioneered the study of factorial *search* designs. He identified the obstacle with many experimental designs that arises from assuming that all higher order effects are negligible when some few are perhaps non-negligible. In the search designs proposed by Srivastava, the goal remains to measure the smaller order interactions, but concurrently to determine whether there are higher order interactions with non-negligible contributions (again under the assumption that a small number of higher order interactions are to be "searched"). Katona and Srivastava [41] lay a mathematical foundation for this in the binary case. Subsequently, there has been substantial research on these search designs; see, for example, [31, 32, 66]. While this research does introduce the notion of location, it is in addition to the measurement afforded by a usual factorial design, and hence asks for stronger properties than we require.

5 Combinatorial Group Testing

Covering arrays are intended to indicate the presence of faults, and designed experiments to measure and model them; neither is defined so as to simply locate them. In a related setting, combinatorial group testing, location of faults has been extensively studied. Du and Hwang [24] provide an excellent textbook discussion of the research on this subject. The basic combinatorial group testing problem is to isolate exactly d, or at most d, defective items among a pool of k items. To do so, one is permitted to *pool* items for testing (i.e. form a subset of the items), and test the pool; if positive then the pool contains one or more defective items, if negative it contains none. Of course, there is the possibility in most real experiments of false outcomes, but we consider the case in which tests on pools are reliable. As noted by Du and Hwang [24], there is an extensive literature on both the *sequential* (or *adaptive*) problem in which a pool is formed only after the outcomes of previous tests are known and on the *nonadaptive* problem in which all pools are assumed to be run in parallel. From origins in testing for diseased individuals in the second world war, group testing was placed on a firm mathematical basis by Sobel and Groll [68]. Applications have been extensive in communications, networking, genomics, and combinatorial search theory. As a result, many different sets of terminology have been developed.

Let M be an $N \times k$ (0,1)-matrix, and $d \leq k$ be an integer. Then M is a *d-separable* matrix if the superposition sum (also called boolean sum or union, which is the componentwise maximum) of any set of d columns of M uniquely determines the set of d columns chosen. It is *$\overline{d}$-separable* if the same holds for all sets of *at most* d columns. In the testing problem, rows represent pools while columns represent items; a '1' in the ith row and jth column indicates that the jth item appears in the ith pool. The superposition sum of the defective columns forms the experimental outcomes, which must correspond to a single "explanation" of the defectives. Naturally any column containing a '1' in a row for which the outcome is '0' cannot be defective, but in general this is not sufficient. Hence a stronger condition is often required. Matrix M is *d-disjunct* if no superposition sum of d columns covers another column.

Treating the columns as binary vectors used as codewords in a communications system, a d-disjunct matrix forms a *d-superimposed code.* Kautz and Singleton [42] pioneered the use of superimposed codes.

Instead treat the rows of M as an N-set, and for each column form a subset containing the indices of the '1' entries in that column, to form a family $\mathcal{F}$ of subsets of an N-set. Then M is d-disjunct if and only if $\mathcal{F}$ is *d-cover-free*; it is $\overline{d}$-separable if and only if $\mathcal{F}$ is *d-union-free*; and it is d-separable if and only if $\mathcal{F}$ is *d-weakly-union-free.* See, for example, [27].

Primarily based on [42], the following relationships obtain among separable and disjunct matrices ([24, Table 7.1]):

$$\begin{tikzcd}
\overline{(d+1)}\text{-separable} \arrow[r, Rightarrow] & d\text{-disjunct} \arrow[r, equal] \arrow[d, Rightarrow, "\text{(delete any row)}"] & \overline{d}\text{-disjunct} \arrow[d, Rightarrow] \\
k\text{-separable}, k < d & d\text{-separable} \arrow[l, Rightarrow] & \overline{d}\text{-separable} \arrow[l, Rightarrow]
\end{tikzcd}$$

For asymptotic results, see [26, 27, 62] and [24, Chap. 7]. Numerous combinatorial constructions are available as well.

There are two fundamental differences between our problem in locating defectives and the underlying hypotheses for disjunct and separable matrices. First and foremost, faults in our environment arise from faulty interactions, not from defective items. Second, we are not free to choose arbitrary pools. Let us consider the first problem first. Torney [74] observed that in certain biological situations, positives arise from interactions among items. He proposed the extension of combinatorial group testing to *sets pooling*, locating specific sets of k items that, when together, form a positive. This has taken the name of *group testing for complexes.* Macula and his colleagues have developed two-stage methods for this problem [48, 49]; however this topic is much less studied than that of defective items.

However the second difference is the more crucial one. In practice the construction of pools often dictate constraints on the size or structure of the pools. For example, pools may be required to be at most a specified size, or an item may be allowed to appear in at most a specified number of pools. Pools may also be constrained to be among those that can be constructed by a simple mechanical process. Our restrictions are somewhat different. Pools are always formed by the selection, for each factor, of exactly one level. While this severely limits the applicability of specific results from the literature on combinatorial group testing to our problem, the basic framework remains quite similar. Hence we believe that there is much value in pursuing the parallels between locating arrays and combinatorial group testing, especially that for complexes.

6 Examples of Locating and Detecting Arrays

In order to illustrate the many definitions, we provide an extensive collection of examples. In each case, we suppose that there are six factors. The first has two levels (0 and 1), while the remaining five have three each (0, 1, and 2). There are $2 \cdot 3^5 = 486$ tests that can be selected; our goal here is to present relatively small sets of tests.

000000
111111
022222

(a)

000000
111111
022222
100000
011111
122222

(b)

000000
001221
010122
121012
122101
112210

(c)

000000
001221
010122
121012
122101
112210
022011
100122

(d)

001221
010122
021012
022101
012210
102112
120211
112021
111202
121120

(e)

000000
101221
110122
121012
022101
012210
102112
120211
112021
111202
121120
021122

(f)

Figure 1: (a) 1-covering array; (b) 1-orthogonal array; (c) (1,1)-locating array; (d) (1,1)-detecting array; (e) (2,1)-locating array; (f) 2-covering array that is not (2,1)-locating.

Figure 1 displays some of the simpler arrays. The array in Figure 1(a) contains every 1-way interaction at least once, and hence is a 1-covering array. It is not (1,1)-locating because interactions (2,1) and (3,1) appear in the same set of rows, in this case just the second row. Nor is it an orthogonal array, because level 0 appears twice for factor 1 while level 1 appears only once. To have equal numbers of occurrences of levels for the first factor, the number of tests must be even; for the remainder, it must be a multiple of 3. Figure 1(b) therefore gives a smallest orthogonal array of strength one, having six rows. It also fails to be (1,1)-locating for the same reason: interactions (2,1) and (3,1) appear in the same set of rows.

Figure 1(c) gives a (1,1)-locating array, also with six tests. It fails to be (1,1)-detecting, because the 1-way interaction (1,0) appears in rows $\{1, 2, 3\}$ while (2,0) appears in $\{1, 2\}$, so the latter interaction is covered by the former. These problems with coverage are remedied in Figure 1(d) by the addition of two rows to form a (1,1)-detecting array. This array is not (2,1)-locating; consider the two 2-way interactions (1,1) and (2,2). Collectively they appear in rows $\{4, 5, 6, 7, 8\}$. But interactions (1,1) and (3,2) appear in the same set of rows. Figure 1(e) presents a (2,1)-locating array. It is not (2,1)-detecting because interactions (1,0) and (1,1) together cover all others. Nor is it a 2-covering array; interaction $\{(2, 0), (3, 0)\}$ is not represented.

000000
101221
010122
021012
022101
012210
102112
120211
112021
111202
121120

(a)

000000
001220
002121
101102
102012
100211
011111
012001
010202
112210
110120
111022
022222
020112
021010
120021
121201
122100

(b)

001221
010122
021012
022101
012210
102112
120211
112021
111202
121120
112002
121200
102120
100212
120021
010220
001022
020102
022010
002201

(c)

000000	110000
021100	122200
011010	122010
101210	002020
112120	100220
011220	111001
102101	001201
020011	121111
112211	001021
100121	022121
010221	102002
120102	012102
012012	000112
110212	021212
120022	111122
022222	

(d)

022212	022002
022110	020022
020100	021202
021200	021011
021101	002222
002211	002120
002102	000010
001021	001122
012022	012001
010020	010111
011220	011120
122222	122220
122201	122001
120210	120011
121210	121001
102100	100220
100020	100021
100102	100110
101001	101112
112122	112101
112112	110211
110000	110120
111212	111022
111102	

(e)

000000	110000
101100	021100
012200	122200
011010	122010
110110	020210
121020	002020
020120	112120
010101	102101
120201	001201
100011	020011
011111	121111
002211	112211
001021	100121
022121	121221
102002	012102
000202	111202
101012	012012
110212	021212
010022	120022
001122	111122
102222	022222

(f)

Figure 2: (a) 2-covering array; (b) 2-orthogonal array; (c) (1,2)-locating array; (d) (1,2)-detecting array; (e) (2,2)-locating array that is not (1,2)-detecting; (f) $(\overline{2}, 2)$-locating array.

Now let us turn to detecting interactions of strength two. Figure 1(f) gives a 2-covering array that is not a (2,1)-locating array; interactions (1,1) and (4,1) appear in the same tests as (1,1) and (6,2). Contrast with Figure 1(e). One is 2-covering but not (2,1)-locating, while the other is (2,1)-locating but not 2-covering. Figure 1(f) is not an optimal 2-covering; instead the test suite in Figure 2(a) is an optimal one, with eleven tests [18]. This array is also (2,1)-locating. It is not (1,2)-locating, however; $\{(1,0),(2,0)\}$ and $\{(1,0),(3,0)\}$ appear in the same tests (only the first in each case). Nor is it an orthogonal array of strength two; indeed an optimal orthogonal array, shown in Figure 2(b), has 18 tests [36].

Now Figure 2(c), with 20 tests, is a (1,2)-locating array but is not (1,2)-detecting. Figure 2(d), with 31 tests, is (1,2)-detecting but not (2,2)-locating. Figure 2(e), with 47 tests, is (2,2)-locating. Despite this it is not $(\overline{2},2)$-locating, nor is it even (1,2)-detecting. However Figure 2(f), with 42 tests, is $(\overline{2},2)$-locating (and hence (2,2)-locating and (1,2)-detecting).

An optimal 3-covering array for these parameters has 33 tests [12], while an optimal 3-orthogonal array has 54 tests [6].

7 Feasibility for Location and Detection

Our goal in providing thumbnail introductions to the three allied areas is to provide a context for the study of locating and detecting arrays. None of these areas offers a solution, but each provides both relevant results and a framework for the combinatorial study of locating and detecting arrays. Naturally the first question is for what parameters a locating or detecting array can exist at all, no matter how large its number of rows. Throughout this discussion we consider an $N \times k$ array of type $(s_1, \dots, s_k)$ with $1 \le s_1 \le s_2 \le \cdots \le s_k$. Write $\tau_t = \sum_{T \subseteq \{1,\dots,k\}, |T|=t} \prod_{i \in T} s_i$, the total number of t-way interactions, and write $\gamma_t = \sum_{i=0}^{t} \tau_t$. Write $\mu_t = \prod_{i=1}^{t} s_i$ and $\nu_t = (\prod_{i=1}^{t-1} s_i) \cdot s_{t+1}$.

We restrict parameters of locating and detecting arrays to nontrivial parameters as follows. First consider the number of faults. Since there are exactly τ_t possible t-way interactions, we treat (d,t)-arrays only when $0 \le d \le \tau_t$. Similarly we treat $(\overline{d},t)$-arrays in the same range; when $d > \tau_t$, a $(\overline{d},t)$-locating array is precisely a $(\overline{\tau_t},t)$-locating array. The same observations apply with $\overline{t}$ in place of t, since the largest independent set of interactions of strength at most t consists of those of strength exactly t.

Since there are exactly γ_t possible interactions of strength at most t, $(d,\hat{t})$-arrays are treated only when $0 \le d \le \gamma_t$. Similarly $(\overline{d},\hat{t})$-locating arrays with $d > \gamma_t$ are precisely $(\overline{\gamma_t},\hat{t})$-arrays.

Next consider the number of columns (factors), k. When $k < t$ and $d > 0$, no $(\overline{d},t)$-locating array can exist, and any $(\overline{d},\overline{t})$-locating array is precisely a $(\overline{d},\overline{k})$-locating array. Hence we treat only cases with $k \ge t$.

Lastly we require that $s_1 > 1$ in order to avoid factors that take on unique levels.

To unify the discussion of locating and detecting arrays, a preliminary observation is useful.

Lemma 7.1. *A (d,t)-detecting array is $(\overline{d},t)$-locating and (d,t)-locating; a $(\overline{d},t)$-locating array is $(d-1,t)$-detecting.*

Proof. An array A that is not (d,t)-locating has two sets $\mathcal{T}_1$ and $\mathcal{T}_2$ of d interactions of strength t for which $\rho(A,\mathcal{T}_1)=\rho(A,\mathcal{T}_2)$. The same sets establish that A is not $(\overline{d},t)$-locating. Now let $T\in\mathcal{T}_2\setminus\mathcal{T}_1$. Then $\rho(A,T)\subseteq\rho(A,\mathcal{T}_1)$, and hence A is not (d,t)-detecting.

For the second statement, if A is not $(d-1,t)$-detecting, consider a set $\mathcal{T}$ of $d-1$ interactions of strength t, and an interaction $T\notin\mathcal{T}$ for which $\rho(A,T)\subseteq\rho(A,\mathcal{T})$. Then $\rho(A,\mathcal{T})=\rho(A,\mathcal{T}\cup\{T\})$, and hence A is not $(\overline{d},t)$-locating. □

Lemma 7.1 holds for $\hat{t}$ in place of t by similar arguments. For $\overline{t}$, the argument is slightly more complicated:

Lemma 7.2. *A $(d,\overline{t})$-detecting array is $(\overline{d},\overline{t})$-locating and $(d,\overline{t})$-locating; a $(\overline{d},\overline{t})$-locating array is $(d-1,\overline{t})$-detecting.*

Proof. An array A that is not $(d,\overline{t})$-locating has two independent sets $\mathcal{T}_1$ and $\mathcal{T}_2$ of d interactions of strength at most t for which $\rho(A,\mathcal{T}_1)=\rho(A,\mathcal{T}_2)$. The same sets establish that A is not $(\overline{d},\overline{t})$-locating. Set $\mathcal{R}_1=\mathcal{T}_1$ and $\mathcal{R}_2=\mathcal{T}_2$. If $\mathcal{R}_2\setminus\mathcal{T}_2$ contains an interaction that is not independent of $\mathcal{T}_2$, remove it. Symmetrically, if $\mathcal{T}_2\setminus\mathcal{R}_2$ contains an interaction that is not independent of $\mathcal{R}_2$, remove it. Repeat until no such dependent interactions remain. If $T\in\mathcal{R}_x\setminus\mathcal{R}_y$ for $\{x,y\}=\{1,2\}$, then $\rho(A,T)\subseteq\rho(A,\mathcal{R}_y)$, and hence A is not $(d,\overline{t})$-detecting. It remains to treat the case that $\mathcal{R}_1=\mathcal{R}_2$; consider the last interaction T that was removed, and suppose that it was removed from $\mathcal{R}_1$. Since T is dependent on some interaction in $\mathcal{R}_2$, and $\mathcal{R}_1=\mathcal{R}_2$, we find that $\mathcal{T}_1$ is not independent, a contradiction.

For the second statement, if A is not $(d-1,\overline{t})$-detecting, consider a set $\mathcal{T}$ of $d-1$ interactions of strength at most t, and an independent interaction $T\notin\mathcal{T}$ for which $\rho(A,T)\subseteq\rho(A,\mathcal{T})$. Then $\rho(A,\mathcal{T})=\rho(A,\mathcal{T}\cup\{T\})$, and hence A is not $(\overline{d},\overline{t})$-locating. □

For detecting arrays, certain cases are equivalent. We employ an auxiliary lemma to explore this.

Lemma 7.3. *For an array A and $s_1\le d\le\tau_t$, there exists a set $\mathcal{T}$ of d independent interactions each of strength at most t, for which $\rho(A,\mathcal{T})$ contains all rows of A.*

Proof. If $s_1\le d\le s_1+\sum_{T\subseteq\{2,\ldots,k\},|T|=t}\prod_{i\in T}s_i$, $\mathcal{T}$ can be chosen to contain all 1-way interactions involving the first factor, together with the remaining t-way interactions, none involving the first factor. In general, for $m\le t$ when $s_1s_2\cdots s_m\le d\le s_1s_2\cdots s_m+\sum_{T\subseteq\{1,\ldots,k\},|T\cap\{1,\ldots,m\}|<m,|T|=t}\prod_{i\in T}s_i$, $\mathcal{T}$ can be chosen to contain all m-way interactions involving the first m factors, together with the remaining t-way interactions, none involving all of the first m factors. Applying this for $m=1,\ldots,t$ yields the result. □

Lemma 7.4. *For $d<\tau_t$, a (d,t)-detecting array is equivalent to a $(\overline{d},t)$-detecting array and a $(d,\overline{t})$-detecting array is equivalent to a $(\overline{d},\overline{t})$-detecting array. For $d<\gamma_t$, a $(d,\hat{t})$-detecting array is equivalent to a $(\overline{d},\hat{t})$-detecting array.*

Proof. For t and $\hat{t}$, if there is a set $\mathcal{T}$ of at most interactions and an interaction $T\notin\mathcal{T}$ for which $\rho(A,T)\subseteq\rho(A,\mathcal{T})$, additional interactions other than T and those in $\mathcal{T}$ can be adjoined to $\mathcal{T}$ without affecting this inclusion. For $\overline{t}$, such additional interactions must be independent. If no interaction of strength at most t is independent of $\mathcal{T}$, then $\rho(A, \mathcal{T})$ contains all rows of A. Lemma 7.3 completes the proof. $\square$

For convenience, we summarize these (together with Lemma 7.6 to follow) in tabular form (when marked with an asterisk, one requires that $d < \gamma_t$; when marked with a plus sign, $d < \tau_t$):

$$\begin{tikzcd}
(\overline{d}, \hat{t})\text{-detecting} \arrow[r, Rightarrow] \arrow[d, Leftrightarrow, "*"] & (\overline{d}, \overline{t})\text{-detecting} \arrow[r, Rightarrow] \arrow[d, Leftrightarrow, "+"] & (\overline{d}, t)\text{-detecting} \arrow[d, Leftrightarrow, "+"] & \\
(d, \hat{t})\text{-detecting} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & (d, \overline{t})\text{-detecting} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & (d, t)\text{-detecting} \arrow[d, Rightarrow] & \\
(\overline{d}, \hat{t})\text{-locating} \arrow[r, Rightarrow] \arrow[d, Rightarrow, "*"] & (\overline{d}, \overline{t})\text{-locating} \arrow[r, Rightarrow] \arrow[d, Rightarrow, "+"] & (\overline{d}, t)\text{-locating} \arrow[r, Rightarrow, "+"] \arrow[d, Rightarrow, "+"] & (d-1, t)\text{-detecting} \\
(d, \hat{t})\text{-locating} \arrow[r, Rightarrow] & (d, \overline{t})\text{-locating} \arrow[r, Rightarrow] & (d, t)\text{-locating} \arrow[r, Rightarrow, "+"] & (d-1, t)\text{-locating}
\end{tikzcd}$$

To establish existence of locating and detecting arrays, it suffices to consider the inclusion of every possible test. The *exhaustive* array is the one having τ_k distinct rows, every possible test.

Lemma 7.5. *When $k = t$, the exhaustive array E is a $(\overline{d}, t)$-detecting array for all d.*

Proof. Every row contains a single interaction. $\square$

7.1 (d, t)- and $(\overline{d}, t)$-Arrays

When $d = \tau_t$, every array is (d, t)-detecting since by hypothesis all of the possible interactions are known to be faulty. So we treat only cases with $d < \tau_t$.

Lemma 7.6. *Every (d, t)-locating array with $\tau_t > d > 0$ is a $(d - 1, t)$-locating array.*

Proof. If $\mathcal{T}$ and $\mathcal{R}$ are different sets of $d-1$ interactions of strength t with $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$, any interaction in neither $\mathcal{R}$ nor $\mathcal{T}$ can be added to both without affecting the equality. It remains to treat the case that $\mathcal{R} \cup \mathcal{T}$ contains all τ_t interactions. When this occurs, their intersection has size at most $d - 2$ while their union has size at least $d + 1$. Therefore $\mathcal{T}$ and $\mathcal{R}$ disagree in at least two interactions. Then let $T \in \mathcal{T} \setminus \mathcal{R}$ and $R \in \mathcal{R} \setminus \mathcal{T}$. Then $\rho(A, \mathcal{R} \cup \{T\}) = \rho(A, \mathcal{T} \cup \{R\})$ but $\mathcal{R} \cup \{T\} \neq \mathcal{T} \cup \{R\}$. $\square$

Lemma 7.7. *If $k > t$ and $min(\mu_t + 1, \nu_t) < d < \tau_t$, no (d, t)-locating array A exists.*

Proof. First we prove the statement for $d = \mu_t + 1$. For the set $\mathcal{T}$ containing all μ_t t-way interactions involving factors $\{1, \ldots, t\}$, $\rho(A, \mathcal{T})$ contains all rows of A. Choose two different t-way interactions that are not in $\mathcal{T}$; adjoining each independently to $\mathcal{T}$ does not alter the rows involved, and hence we have produced two different sets of $\mu_t + 1$ t-way interactions generating the same set of rows.

Next consider the case when $\nu_t = \mu_t$. Form the set $\mathcal{R}$ of all ν_t t-way interactions involving factors $\{1, \ldots, t-1\} \cup \{t+1\}$. Then $\rho(A, \mathcal{R})$ contains all rows of A. Then $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$ but $\mathcal{R} \neq \mathcal{T}$.

Complete the proof by applying Lemma 7.6. $\square$

In fact a much stronger bound holds:

Lemma 7.8. *If $k > t$ and $min(s_1, s_2 - 1) < d \leq \mu_t$, no (d,t)-locating array A exists.*

Proof. We first prove the statement for $d = s_1 + 1$. Consider the t-way interactions $T = \{(2, v_{21}), \ldots, (t+1, v_{t+1,1})\}$ and $R = \{(2, v_{21}), \ldots, (t+1, v_{t+1,2})\}$. Form the set $\mathcal{T}$ of t-way interactions $\{\{(2, v_{21}), \ldots, (t, v_{t1})\} \cup \{(1, i)\} : i \in S_1\}$. Then $\rho(A, T) \subseteq \rho(A, \mathcal{T})$ and $\rho(A, R) \subseteq \rho(A, \mathcal{T})$, and hence $\rho(A, \mathcal{T} \cup \{R\}) = \rho(A, \mathcal{T} \cup \{T\})$.

Next we consider the case when $d = s_1 = s_2$. Form $\mathcal{T}$ as $\{\{(3, v_{31}), \ldots, (t+1, v_{t+1,1})\} \cup \{(1, i)\} : i \in S_1\}$ and $\mathcal{R}$ as $\{\{(3, v_{31}), \ldots, (t+1, v_{t+1,1})\} \cup \{(2, i)\} : i \in S_2\}$. Then $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$ but $\mathcal{R} \neq \mathcal{T}$. ☐

Lemma 7.9. *Every $(\overline{d}, t)$-locating array is (d', t)-locating for every $d' \leq min(d, \tau_t)$.*

Proof. This follows immediately from the definitions. ☐

Lemma 7.10. *When $k > t \geq 1$ and $0 < d < min(s_1 + 1, s_2)$, the exhaustive array E is $(\overline{d}, t)$-locating and hence is a (d, t)-locating array.*

Proof. Consider two different sets $\mathcal{T}$ and $\mathcal{R}$, each of at most s_1 t-way interactions, with $\mathcal{R}$ having at least as many interactions as $\mathcal{T}$. If $\mathcal{T}$ has fewer than s_1 interactions, consider an interaction R in $\mathcal{R} \setminus \mathcal{T}$. For every $T \in \mathcal{T}$, if R and T are on the same factors, then the sets of rows in which they occur are disjoint. Otherwise if ℓ is a factor of T but not R, T can contain at most $\frac{1}{s_\ell}$ of the rows containing R. Since $|\mathcal{T}| < |\mathcal{R}| \leq d \leq s_1$, $\rho(E, R) \not\subseteq \rho(E, \mathcal{T})$.

Otherwise $\mathcal{T}$ and $\mathcal{R}$ each have s_1 interactions. Choose $T \in \mathcal{T} \setminus \mathcal{R}$ and $R \in \mathcal{R} \setminus \mathcal{T}$. Every interaction in $\mathcal{T}$ covers at most $\frac{1}{s_1}$ rows in $\rho(E, R)$, and symmetrically every interaction in $\mathcal{R}$ covers at most $\frac{1}{s_1}$ rows in $\rho(E, T)$. Thus all rows can be covered when each accounts for *exactly* $\frac{1}{s_1}$. Consider the factors shared by T and R. If they are on the same factors, then they cover disjoint rows. Let α be a factor of T but not R, and β be a factor of R but not T. Then R covers at most $\frac{1}{s_\beta}$ of those covered by T, and T covers at most $\frac{1}{s_\alpha}$ of those covered by R. This necessitates that $s_\alpha = s_\beta = s_1$, and hence since $\alpha \neq \beta$ that $s_1 = s_2$. Since $d \leq s_2 - 1$ by hypothesis, $\mathcal{T}$ and $\mathcal{R}$ do not arise in the same sets of rows. ☐

Lemma 7.11. *When $k > t$ and $\tau_t > d \geq s_1$, there is no (d, t)-detecting array.*

Proof. Form the set $\mathcal{T}$ of t-way interactions $\{\{(2, v_{21}), \ldots, (t, v_{t1})\} \cup \{(1, i)\} : i \in S_1\}$. Then for $T = \{(2, v_{21}), \ldots, (t, v_{t1}), (t+1, v_{t+1,1})\}$, $\rho(A, T) \subseteq \rho(A, \mathcal{T})$. ☐

Lemma 7.12. *When $d < s_1$, the exhaustive array is a (d, t)-detecting array.*

Proof. This follows from Lemmas 7.1 and 7.10 except when $d + 1 = s_1 = s_2$, so we treat this case. If $\mathcal{T}$ is a set of fewer than $s_1 = s_2$ t-way interactions and $T \notin \mathcal{T}$ is a t-way interaction, every interaction in $\mathcal{T}$ accounts for at most $\frac{1}{s_1}$ of the tests containing T; hence some test containing T contains no interaction in $\mathcal{T}$. ☐

7.2 $(d, \overline{t})$- and $(\overline{d}, \overline{t})$-Arrays

Lemma 7.13. *Every $(d, \overline{t})$-locating array with $d < \tau_t$ is (d', t')-locating for every $t' \leq t$ and $d' \leq min(d, \tau_{t'})$.*

Proof. Suppose to the contrary that A is $(d, \overline{t})$-locating but not (d', t')-locating. Then there are sets $\mathcal{R}$ and $\mathcal{T}$, each containing d' independent interactions of strength t', for which $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$. We can adjoin further independent interactions to enlarge each (to get d interactions), unless in the process we find that $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$ contains all rows. Lemma 7.3 completes the proof. □

Lemma 7.14. *Every $(\overline{d}, \overline{t})$-locating array is $(\overline{d}, t')$-locating for every $t' \leq t$ and $(d', \overline{t})$-locating for every $d' \leq min(d, \tau_t)$.*

Proof. This follows immediately from the definitions. □

Lemma 7.15. *If $s_1 \leq d$, no $(\overline{d}, \overline{t})$-locating array A exists.*

Proof. When $d = s_1$, form set $\mathcal{T}$ to contain the unique 0-way interaction. Form $\mathcal{R}$ to contain all 1-way interactions involving the first factor. Then $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$ but $\mathcal{R} \neq \mathcal{T}$. □

Lemma 7.16. *When $t \geq 1$ and $0 \leq d < s_1$, the exhaustive array E is a $(\overline{d}, \overline{t})$-detecting array.*

Proof. Let $\mathcal{T}$ be an independent set of at most d interactions of strength at most t. We show that for every $T \notin \mathcal{T}$ that is independent of $\mathcal{T}$, there is a test containing T but no interaction of $\mathcal{T}$. Consider a specific $T' \in \mathcal{T}$; if the factors in T' all appear in T, or those in T all appear in T', the sets of rows in which they arise are disjoint. So let ℓ be a factor in T' not in T. The rows containing T admit every one of the s_ℓ possible levels for factor ℓ, and hence the rows containing T' account for at most $\frac{1}{s_1}$ of those containing T. Then $\rho(E, T)$ is not contained in $\rho(E, \mathcal{T})$. □

Lemma 7.17. *When $t \geq 1$, $k \geq 2$, and $d = s_1 < s_2$, the exhaustive array E is a $(d, \overline{t})$-locating array.*

Proof. Since $s_2 > d = s_1 \geq 2$ and sets of interactions are required to be independent, we need only consider sets that do not contain the 0-way interaction. Then the proof parallels that of Lemma 7.10. □

7.3 $(\overline{d}, \hat{t})$-Arrays

Lemma 7.18. *Every $(d, \hat{t})$-locating array is $(d, \overline{t})$-locating when $d < \tau_t$, and every $(\overline{d}, \hat{t})$-locating array is $(\overline{d}, \overline{t})$-locating.*

Lemma 7.19. *When $0 < d < \gamma_t$, no $(d, \hat{t})$-detecting array exists.*

Proof. The 0-way interaction covers all others. □

Lemma 7.20. *If $2 \leq d < \gamma_t$, no $(d, \hat{t})$-locating array exists.*

Proof. Form set $\mathcal{T}$ to contain the unique 0-way interaction and $d-1$ others. Form set $\mathcal{R}$ to contain the 0-way interaction and $d-1$ others, not identical to those chosen earlier. Then $\rho(A, \mathcal{R}) = \rho(A, \mathcal{T})$ but $\mathcal{R} \neq \mathcal{T}$. $\square$

Let us now dispense with some trivial cases. When $d = 0$, any array is a $(\overline{d}, \hat{t})$-locating array for all t, including the array with no rows at all! So suppose that $d \geq 1$. When $t = 0$, $d \leq 1$. Every array with at least one row is $(\overline{1}, \overline{0})$-locating (there is, after all, only one interaction to consider). So suppose henceforth that $t \geq 1$.

When $d = \gamma_t$, every array is $(d, \hat{t})$-locating since all interactions of strength at most t are known to be faulty. It remains to treat cases with $0 < d < 2$ (that is, $d = 1$) and $k \geq t \geq 1$ for $(d, \hat{t})$-locating arrays, in view of Lemma 7.20.

Lemma 7.21. *When $k \geq t \geq 1$ and $d \in \{0, 1\}$, the exhaustive array E is $(\overline{d}, \hat{t})$-locating and hence is a $(d, \hat{t})$-locating array.*

Proof. Consider different interactions T and R, each of strength at most t. There is at least one row that contains one but not the other. $\square$

7.4 Summary

Type of Array		Constraint	Lemmata
$(d,\ t)$-	locating	$d < \min(s_1 + 1, s_2)$ or $d = \tau_t$ if $k > t$	7.8, 7.10
		$d \leq \tau_t$ if $k = t$	7.5
	detecting	$d < s_1$ or $d = \tau_t$ if $k > t$	7.11, 7.12
		$d \leq \tau_t$ if $k = t$	7.5
$(\overline{d},\ t)$-	locating	$d < \min(s_1 + 1, s_2)$ if $k > t$	7.8, 7.10
		any d if $k = t$	7.5
	detecting	$d < s_1$ if $k > t$	7.4
		any d if $k = t$	7.5
$(d,\ \overline{t})$-	locating	$d < \min(s_1 + 1, s_2)$ or $d = \tau_t$	7.8, 7.10, 7.17
	detecting	$d < s_1$ or $d = \tau_t$	7.16
$(\overline{d},\ \overline{t})$-	locating	$d < s_1$	7.15, 7.16
	detecting	$d < s_1$	7.4, 7.16
$(d,\ \hat{t})$-	locating	$d \in \{0, 1, \gamma_t\}$	7.20, 7.21
	detecting	$d \in \{0, \gamma_t\}$	7.19
$(\overline{d},\ \hat{t})$-	locating	$d \in \{0, 1\}$	7.20, 7.21
	detecting	$d = 0$	7.4

It seems that placing just an upper bound on the strength, without requiring independence, leads to a relatively uninteresting problem (the $\hat{t}$ case). At most one fault can be located, and then only knowing *a priori* that at most one fault is present. It would be a mistake to dismiss this problem so quickly, however. Assume that $k \geq t$. Using any t-covering array, one can examine from the outcomes whether there is no faulty interaction (no tests fail), or whether the 0-way interaction is faulty (all tests fail), or whether some interaction of strength greater than 0 is faulty (some but not all tests fail). In the latter case, *which* interaction is faulty? In order to distinguish, we require that for any two interactions T_1, T_2 of strength at most t, $\rho(A, T_1) \neq \rho(A, T_2)$. Now if the strengths of T_1 and T_2 disagree, choose without loss of generality a row $r \in \rho(A, T_1) \setminus \rho(A, T_2)$. Extend T_1 to a t-way interaction using levels in row r. Extend T_2 to a t-way interaction arbitrarily. The inequality $\rho(A, T_1) \neq \rho(A, T_2)$ is preserved. So it suffices to require the inequality for every two interactions of strength equal to t. A $(\overline{1}, \hat{t})$-locating array is also a $(1, t)$-locating array and a t-covering array; the connection with covering arrays becomes crucial in the next section.

8 Logarithmic Growth

We have characterized the cases in which a locating array can exist, but it tells us little about the number of tests needed. The *size* of an array is its number of rows; size directly determines testing cost by determining the number of tests to be executed. In this section we establish that for fixed d and t and fixed maximum number of levels per factor, the size of a minimum detecting array grows logarithmically in the number of factors. This stands in stark contrast to the analogous situation for designed experiments, and relies on parallel results for covering arrays. We begin by treating the uniform case in which every factor has the same number of levels.

Theorem 8.1. *Every $(\overline{d}, t)$-detecting array with $s_1 = \cdots = s_k = d + 1$ is a $(t + 1)$-covering array.*

Proof. Let A be an $N \times k$ array in which every column contains $d + 1$ symbols that is not $(t + 1)$-covering. Let $T = \{(f_i, \sigma_i) : 1 \leq i \leq t + 1\}$ be a $(t + 1)$-way interaction that is not covered. Form $\mathcal{T} = \{\{(f_i, \sigma_i) : 1 \leq i < t\} \cup \{(f_{t+1}, s)\} : s \in S_{f_{t+1}} \setminus \{\sigma_{t+1}\}\}$. Let $T' = T \setminus \{(f_{t+1}, \sigma_{t+1}\}$. Then $\rho(A, T') \subseteq \rho(A, \mathcal{T})$ and hence A is not (d, t)-detecting. $\square$

One might hope that the converse also holds. Consider the following two covering arrays of strength two:

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

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

The array on the left is indeed a $(\overline{2}, 1)$-detecting array. Although larger, the array on the right is not. In fact the rows containing the first factor at level 0 are covered by the union of those containing one of the second or third factors at level 0. Indeed while covering $(t+1)$-way interactions is necessary, it is far from sufficient:

Theorem 8.2. *There exists a* $\mathrm{CA}(N; t+d-1, k, v)$ *for* $d < v$ *and* $k \geq d+t$ *that is not* (d, t)*-detecting.*

Proof. Let A be an arbitrary $\mathrm{CA}(N; t+d-1, k, v)$. Identify $d+1$ sets $F_1, \dots, F_{d+1}$ each containing exactly t factors, so that each contains the first $t-1$ factors along with one other. First, for every row of A in which every entry in F_{d+1} equals 0, form t rows by replacing each 0 in turn by a 1. Then the t-way interaction $T = \{(f, 0) : f \in F_{d+1}\}$ is uncovered. Now we construct a large collection of additional rows; each contains T. For $1 \leq i \leq d$, we select a 0 level on the factor in $F_i \setminus F_{d+1}$; then for every $1 \leq j \leq d$ with $j \neq i$, we select an arbitrary nonzero levels for the factor in $F_j \setminus F_{d+1}$, and finally we select arbitrary levels for the remaining $k - (d+t)$ factors. In this way $d(v-1)^{d-1} v^{k-(d+t)}$ tests are added. It is easy to verify that the result is another $\mathrm{CA}(N'; t+d-1, k, v)$ A'. Now let $\mathcal{T} = \{\{(f, 0) : f \in F_i\} : 1 \leq i \leq d\}$. Then $\rho(A', T) \subseteq \rho(A', \mathcal{T})$. □

Despite this, some covering arrays of strength $t+1$ do provide detecting arrays. Consider a specific t-way interaction T. Suppose that in the $(t+1)$-covering array it appears λ times (evidently $\lambda \geq d+1$ since every $(t+1)$-way interaction containing T must appear, but λ may be much larger). For each factor f not appearing in T, let μ_f be the maximum over $s \in S_f$ of the number of rows in which the $(t+1)$-way interaction $T \cup \{(f, s)\}$ appears. For factor f of T let $\mu_f = 0$. In the example on the left above with $t = 1$, $\lambda = 3$ and $\mu_f = 1$ (for all choices). However in the example on the right, for $T = \{(0, 0)\}$, we find that $\lambda = 4$ and $\mu_1 = \mu_2 = 2$. This accounts for the array on the left being $(\overline{2}, 1)$-detecting while explaining in part why the array on the right is not. Using these counts we can provide a sufficient condition for detection:

Theorem 8.3. *Let* A *be a* $\mathrm{CA}(N; t+1, k, d+1)$*. For the* t*-way interaction* T*, suppose that it appears* λ *times, and for every factor* f *not in* T*, the* μ_f *values are determined as above. Then suppose that for every interaction* T *and every set* D *of* d *factors,* $\lambda > \sum_{f \in D} \mu_f$*. Then* A *is an* $(\overline{d}, t)$*-detecting array.*

Proof. Interaction T appears in λ rows, and any row containing T and a level for a further factor f accounts for at most μ_f of these. □

At first glance this appears to provide a useful converse to Theorem 8.1. However, the limitation on the number d of faulty interactions to be detected can be quite severe, depending on the structure of the covering array A. Nevertheless, if we suppose that the covering array has the much stronger property that every $(t+1)$-way interaction occurs the same number of times, then we find that for every t-way interaction T occurring λ times, $\mu_f = \lambda/(d+1)$ (always). Then the hypotheses of Theorem 8.3 are trivially met.

A covering array $\mathrm{CA}(N; t, k, v)$ in which every t-way interaction occurs the same number (N/v^t) times, is an *orthogonal array.* These have been extensively studied [36] from a number of perspectives, not least because they underlie fractional factorial designs. However from our viewpoint a disappointment is in store. The well-known Rao bounds [36, Theorem 2.1] show that orthogonal arrays require a large number of rows:

Theorem 8.4. *An orthogonal array with k factors each with v levels, having strength $t \geq 2$, has at least N rows where*

$$N = \begin{cases} \sum_{i=0}^{u} \binom{k}{i}(v-1)^i & \text{if } t = 2u \\ \sum_{i=0}^{u} \binom{k}{i}(v-1)^i + \binom{k-1}{u}(v-1)^{u+1} & \text{if } t = 2u+1 \end{cases}$$

The disappointment is that, while orthogonal arrays balance occurrence of $(t+1)$-way interactions, and hence lead to (d,t)-detecting arrays, the number of rows required grows as a polynomial function of k for fixed strength and number of levels. In fact it is consistent with the use of orthogonal arrays as factorial designs for estimation rather than mere location. This is unlike the situation for covering arrays, where the number of rows grows logarithmically in k.

Perhaps we are asking too much when we require the balance of an orthogonal array. Theorem 8.3 asks for an "approximately" balanced occurrence of $(t+1)$-subsets, but does not require equality. It is time for a positive result:

Theorem 8.5. *Every* $\mathrm{CA}(N; t+d, k, v)$ *with $d < v$ is a (d,t)-detecting array. Indeed for $d < s_1 \leq s_2 \leq \cdots \leq s_k$, every MCA$(N; t+d, (s_1 \cdots s_k))$ is a (d,t)-detecting array.*

Proof. Let T be a t-way interaction, and consider a set $\mathcal{T}$ of d other t-way interactions. If any $T' \in \mathcal{T}$ involves the same factors as T, but $T' \neq T$, then the rows in which they appear are disjoint. So suppose that no interaction of $\mathcal{T}$ involves exactly the same t factors as T. Choose a set F of $d' \leq d$ factors not in T so that every $T' \in \mathcal{T}$ involves at least one factor of F; this can be done since $\mathcal{T}$ contains d interactions each having a factor not contained in T. Now for every $f \in F$, choose a level σ_f that appears in no interaction of $\mathcal{T}$ as the level for factor f; this can be done because factor f has more than d levels but at most d appear in interactions of $\mathcal{T}$. Then form the $(t+d')$-way interaction T' by adjoining to T the selections $\{(f, \sigma_f) : f \in F\}$. Now T' is covered in some row of every $\mathrm{CA}(N; t+d, k, d+1)$, and yet this row contains none of the interactions in $\mathcal{T}$ by construction. □

This appears to be very weak, but it leads to a surprising result:

Theorem 8.6. *For fixed d, t, and m, the number of rows N required in a $(\overline{d}, t)$-detecting array with k factors each having at least $d+1$ and at most m levels grows logarithmically in k.*

Proof. The logarithmic growth applies for $\mathrm{CA}(N; t+d, k, m)$s [58, 59]. Apply Theorem 8.5. □

Contrast this with the polynomial growth of N as a function of k for orthogonal arrays (Theorem 8.4). How does one account for this dramatic difference? Orthogonal arrays distinguish all subsets of interactions, but detecting arrays only distinguish small subsets of interactions. While the total number of subsets grows exponentially in the number of factors, the number of subsets of some fixed size d grows only polynomially in k.

9 Conclusions

The nonadaptive location of interaction faults in a component-based system has motivated the introduction of new classes of combinatorial arrays, detecting and locating arrays, to form fault interaction test suites that permit the location of a specified number of faults of specified strength. The similarities and differences with covering arrays, designed experiments, and pooling designs are outlined. Finally basic constraints on feasibility have been presented.

The study of locating and detecting arrays poses challenges in each of the areas outlined. While covering arrays have been studied with no need to distinguish among interactions, and designed experiments studied with the need to distinguish all interactions of a specified strength, detection and location form a middle ground. Of course one wants to combine the ability of covering arrays to provide small test suites with the ability of designed experiments to distinguish the interactions; techniques from both areas appear to be relevant here. Perhaps the only mathematical model to follow in determining a specified number of faults is afforded by combinatorial group testing. While fault detection falls naturally into group testing for complexes, it limits the structure of complexes to be located in a significant way. More importantly, the construction of pools in this variant of group testing for complexes deviates substantially from those previously studied. It is premature to treat the fault location problems merely as a special case of any one of these three established areas, and we believe that each contains tools of value in exploring the location of interaction faults.

We have determined the basic necessary conditions for detecting and locating arrays to exist when no limit is placed on the number of tests to be performed. Once feasibility is determined, the natural goal is to minimize the size of the test suite constructed. Here the contrast between covering arrays and designed experiments is the most stark: As the number of factors k increases, the number of tests in a covering array grows logarithmically in k while for a designed experiment it grows polynomially in k. We establish that, in this regard, detecting and locating arrays exhibit the behaviour of covering arrays; the number of tests grows logarithmically in k.

Although the asymptotic growth rate is determined, the construction of specific locating and detecting arrays with fewest tests appears to be a challenging problem. However, the close connections with covering arrays, pooling designs, and designed experiments provide a wealth of techniques, both combinatorial and computational, for beginning to explore these construction problems.

Acknowledgments

Thanks to Violet Syrotiuk, Doug Montgomery, Myra Cohen, Dean Hoskins, and Gyula Katona for helpful discussions about this research. Thanks to the referees for helpful comments.

References

[1] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3:289–304, 1992. Addendum in *Random Structures and Algorithms* 4 (1993), 119–120.

[2] J. Azar, R. Motwani, and J. Naor. Approximating probability distributions using small sample spaces. *Combinatorica*, 18:151–171, 1998.

[3] J. Bierbrauer and H. Schellwat. Almost independent and weakly biased arrays: efficient constructions and cryptologic applications. *Lecture Notes in Computer Science*, 1880:533–543, 2000.

[4] S. Bisgaard. A method for the identification of defining contrasts for $2^k - p$ designs. *Journal of Quality Technology*, 25:28–35, 1993.

[5] S. Bisgaard. A comparative analysis of the performance of Taguchi's linear graphs for the design of two-level fractional factorials. *Applied Statistics*, 45:311–322, 1996.

[6] A. E. Brouwer, A. M. Cohen, and M. V. M. Nguyen. Orthogonal arrays of strength 3 and small run sizes. *J. Stat. Plann. Infer.*, 136:3268–3280, 2006.

[7] R. Brownlie, J. Prowse, and M. S. Phadke. Robust testing of AT&T PMX/StarMAIL using OATS. *AT&T Technical Journal*, 71:41–47, 1992.

[8] R. C. Bryce and C. J. Colbourn. The density algorithm for pairwise interaction testing. *Software Testing, Verification, and Reliability*, to appear.

[9] K. Burr and W. Young. Combinatorial test techniques: Table-based automation, test generation, and code coverage. In *Proceedings of the International Conference on Software Testing Analysis and Review*, pages 503–513, New York, 1998. ACM.

[10] J. N. Cawse. Experimental design for combinatorial and high throughput materials development. *GE Global Research Technical Report*, 29(9):769–781, 2002.

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

[12] M. A. Chateauneuf, C. J. Colbourn, and D. L. Kreher. Covering arrays of strength 3. *Des. Codes Crypt.*, 16:235–242, 1999.

[13] D. M. Cohen, S. R. Dalal, J. Parelius, and G. C. Patton. The combinatorial design approach to automatic test generation. *IEEE Software*, 13:82–88, 1996.

[14] G. Cohen, S. Litsyn, and G. Zémor. On greedy algorithms in coding theory. *IEEE Trans. Inform. Theory*, 42:2053–2057, 1996.

[15] M. B. Cohen. *Designing test suites for software interaction testing.* PhD thesis, The University of Auckland, Department of Computer Science, 2004.

[16] M. B. Cohen, C. J. Colbourn, P. B. Gibbons, and W. B. Mugridge. Constructing test suites for interaction testing. In *Proceedings of the International Conference on Software Engineering (ICSE 2003)*, pages 38–48, Los Alamitos, CA, 2003. IEEE.

[17] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling. Constructing strength three covering arrays with augmented annealing. *Discrete Math.*, to appear.

[18] C. J. Colbourn. Combinatorial aspects of covering arrays. *Le Matematiche (Catania)*, 58:121–167, 2004.

[19] C. J. Colbourn. Strength two covering arrays: Existence tables and projection. *Discrete Math.*, to appear.

[20] C. J. Colbourn and J. H. Dinitz, editors. *The CRC Handbook of Combinatorial Designs.* Chapman and Hall/CRC, Boca Raton, FL, second edition, 2007.

[21] C. J. Colbourn, S. S. Martirosyan, G. L. Mullen, D. E. Shasha, G. B. Sherwood, and J. L. Yucas. Products of mixed covering arrays of strength two. *Journal of Combinatorial Designs*, 14:124–138, 2006.

[22] C. J. Colbourn, S. S. Martirosyan, Tran van Trung, and R. A. Walker II. Roux-type constructions for covering arrays of strengths three and four. *Designs Codes Crypt.*, 41:33–57, 2006.

[23] S. R. Dalal, A. J. N. Karunanithi, J. M. L. Leaton, G. C. P. Patton, and B. M. Horowitz. Model-based testing in practice. In *Proc. Intl. Conf. on SoftwareEngineering,(ICSE '99)*, pages 285–294, 1999.

[24] D.-Z. Du and F. K. Hwang. *Combinatorial group testing and its applications.* World Scientific Publishing Co. Inc., River Edge, NJ, second edition, 2000.

[25] S. Dunietz, W. K. Ehrlich, B. D. Szablak, C. L. Mallows, and A. Iannino. Applying design of experiments to software testing. In *Proc. Intl. Conf. on Software Engineering (ICSE '97)*, pages 205–215, Los Alamitos, CA, 1997. IEEE.

[26] A. G. D′yachkov, V. V. Rykov, and A. M. Rashad. Superimposed distance codes. *Problems Control Inform. Theory*, 18:237–250, 1989.

[27] P. Erdős, P. Frankl, and Z. Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel J. Math.*, 51:79–89, 1985.

[28] L. Gargano, J. Körner, and U. Vaccaro. Sperner theorems on directed graphs and qualitative independence. *Journal of Combinatorial Theory (A)*, 61:173–192, 1992.

[29] L. Gargano, J. Körner, and U. Vaccaro. Sperner capacities. *Graphs and Combinatorics*, 9:31–46, 1993.

[30] L. Gargano, J. Körner, and U. Vaccaro. Capacities: from information to extremal set theory. *Journal of Combinatorial Theory (A)*, 68:296–316, 1994.

[31] S. Ghosh and C. Burns. Two general classes of search designs for factor screening experiments with factors at three levels. *Metrika*, 54:1–17, 2001.

[32] S. Ghosh and C. Burns. Comparison of four new general classes of search designs. *Austral. New Zealand J. Stat.*, 44:357–366, 2002.

[33] A. P. Godbole, D. E. Skipper, and R. A. Sunley. t-covering arrays: upper bounds and poisson approximations. *Combinatorics, Probability and Computing*, 5:105–118, 1996.

[34] A. Hartman. Software and hardware testing using combinatorial covering suites. In M. C. Golumbic and I. B.-A. Hartman, editors, *Interdisciplinary Applications of Graph Theory, Combinatorics, and Algorithms*, pages 237–266. Springer, Norwell, MA, 2005.

[35] A. Hartman and L. Raskin. Problems and algorithms for covering arrays. *Discrete Math.*, 284:149–156, 2004.

[36] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays.* Springer-Verlag, New York, 1999.

[37] B. Hnich, S. Prestwich, and E. Selensky. Constraint-based approaches to the covering test problem. *Lecture Notes in Computer Science*, 3419:172–186, 2005.

[38] D. S. Hoskins, C. J. Colbourn, and M. Kulahci. Sub D-optimal designs for screening experiments. *Amer. J. Math. Manag. Sci.*, to appear.

[39] D. S. Hoskins, C. J. Colbourn, and D. C. Montgomery. Software performance testing using covering arrays. In *Fifth International Workshop on Software and Performance (WOSP 2005)*, pages 131–137, 2005.

[40] G. Katona. Two applications (for search theory and truth functions) of Sperner type theorems. *Periodica Math.*, 3:19–26, 1973.

[41] G. Katona and J. N. Srivastava. Minimal 2-coverings of a finite affine space based on GF(2). *J. Stat. Plann. Infer.*, 8:375–388, 1983.

[42] W. H. Kautz and R. R. Singleton. Nonrandom binary superimposed codes. *IEEE Trans. Inform. Theory*, 10:363–377, 1964.

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

[44] J. Körner and M. Lucertini. Compressing inconsistent data. *IEEE Trans. Information Theory*, 40:706–715, 1994.

[45] D. R. Kuhn and M. Reilly. An investigation of the applicability of design of experiments to software testing. In *Proc. 27th Annual NASA Goddard/IEEE Software Engineering Workshop*, pages 91–95, Los Alamitos, CA, 2002. IEEE.

[46] D. R. Kuhn, D. R. Wallace, and A. M. Gallo. Software fault interactions and implications for software testing. *IEEE Trans. Software Engineering*, 30(6):418–421, 2004.

[47] K. Kurosawa, T. Johansson, and D. R. Stinson. Almost k-wise independent sample spaces and their cryptologic applications. *Lecture Notes in Computer Science*, 1233:409–421, 1997.

[48] A. J. Macula, V. V. Rykov, and S. Yekhanin. Trivial two-stage group testing for complexes using almost disjunct matrices. *Discrete Appl. Math.*, 137(1):97–107, 2004.

[49] A. J. Macula, D. C. Torney, and P. A. Vilenkin. Two-stage group testing for complexes in the presence of errors. In *Discrete mathematical problems with medical applications (New Brunswick, NJ, 1999)*, pages 145–157. Amer. Math. Soc., Providence, RI, 2000.

[50] R. Mandl. Orthogonal latin squares: An application of experiment design to compiler testing. *Communications of the ACM*, 28(10):1054–1058, 1985.

[51] E. Marczewski. Independence d'ensembles et prolongement de mesures. *Colloq. Math.*, 1:122–132, 1948.

[52] S. S. Martirosyan and C. J. Colbourn. Recursive constructions for covering arrays. *Bayreuther Math. Schriften*, 74:266–275, 2005.

[53] S. S. Martirosyan and Tran van Trung. On t-covering arrays. *Des. Codes Cryptogr.*, 32:323–339, 2004.

[54] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, Inc., 6 edition, 2005.

[55] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Computing*, 22:838–856, 1993.

[56] K. Nurmela. Upper bounds for covering arrays by tabu search. *Discrete Applied Mathematics*, 138:143–152, 2004.

[57] M. S. Phadke. Planning efficient software tests. *CrossTalk - Journal of Defense Software Engineering*, 10:11–15, 1997.

[58] S. Poljak, A. Pultr, and V. Rödl. On qualitatively independent partitions and related problems. *Discrete Applied Math.*, 6:193–205, 1983.

[59] S. Poljak and Z. Tuza. On the maximum number of qualitatively independent partitions. *Journal of Combinatorial Theory (A)*, 51:111–116, 1989.

[60] A. Réyni. *Foundations of Probability.* Wiley, New York, 1971.

[61] G. Roux. *k-Propriétés dans les tableaux de n colonnes: cas particulier de la k-surjectivité et de la k-permutivité.* PhD thesis, Université de Paris, 1987.

[62] M. Ruszinkó. On the upper bound of the size of the r-cover-free families. *J. Combin. Theory Ser. A*, 66:302–310, 1994.

[63] H. Scheffe. A method for judging all contrasts in the analysis of variance. *Biometrika*, pages 87–104, 1953.

[64] G. Seroussi and N. H. Bshouty. Vector sets for exhaustive testing of logic circuits. *IEEE Transactions on Information Theory*, 34:513–522, 1988.

[65] D. E. Shasha, A. Y. Kouranov, L. V. Lejay, M. F. Chou, and G. M. Coruzzi. Using combinatorial design to study regulation by multiple input signals: A tool for parsimony in the post-genomics era. *Plant Physiology*, 127:1590–1594, 2001.

[66] T. Shirakura, T. Takahashi, and J. N. Srivastava. Searching probabilities for nonzero effects in search designs for the noisy case. *Ann. Statist.*, 24:2560–2568, 1996.

[67] N. J. A. Sloane. Covering arrays and intersecting codes. *Journal of Combinatorial Designs*, 1:51–63, 1993.

[68] M. Sobel and P. A. Groll. Group testing to eliminate efficiently all defectives in a binomial sample. *Bell System Tech. J.*, 38:1179–1252, 1959.

[69] J. N. Srivastava. Designs for searching non-negligible effects. In J. N. Srivastava, editor, *A Survey of Statistical Design and Linear Models*, pages 507–519. North–Holland, 1975.

[70] B. Stevens. *Transversal Covers and Packings.* PhD thesis, Mathematics, University of Toronto, 1998.

[71] G. Taguchi and Y. Wu. *Introduction to Off-Line Quality Control.* Central Japan Quality Control Association, 1985.

[72] K. C. Tai and L. Yu. A test generation strategy for pairwise testing. *IEEE Transactions on Software Engineering*, 28:109–111, 2002.

[73] D. T. Tang and C. L. Chen. Iterative exhaustive pattern generation for logic testing. *IBM Journal Research and Development*, 28:212–219, 1984.

[74] D. C. Torney. Sets pooling designs. *Ann. Comb.*, 3(1):95–101, 1999.

[75] Y. W. Tung and W. S. Aldiwan. Automating test case generation for the new generation mission software system. In *Proc. 30th IEEE Aerospace Conference*, pages 431–437, Los Alamitos, CA, 2000. IEEE.

[76] R. A. Walker II and C. J. Colbourn. Perfect hash families: Constructions and existence. *J. Math. Crypt.*, to appear.

[77] R. A. Walker II and C. J. Colbourn. Tabu search for covering arrays using permutation vectors. *J Stat. Plann. Infer.*, to appear.

[78] A. W. Williams and R. L. Probert. A measure for component interaction test coverage. In *Proc. ACS/IEEE Intl. Conf. on Computer Systems and Applications*, pages 301–311, Los Alamitos, CA, 2001. IEEE.