

The Test Suite Generation Problem: Optimal Instances and Their Implications

Christine T. Cheng
Department of Computer Science
University of Wisconsin–Milwaukee, Milwaukee, WI 53211, USA.
ccheng@cs.uwm.edu

Abstract

In the *test suite generation problem* (TSG) for software systems, \mathcal{I} is a set of n input parameters where each $I \in \mathcal{I}$ has $\kappa(I)$ data values, and \mathcal{O} is a collection of subsets of \mathcal{I} where the interactions of the parameters in each $O \in \mathcal{O}$ are thought to affect the outcome of the system. A test case for $(\mathcal{I}, \mathcal{O}, \kappa)$ is an n -tuple (t_1, t_2, \dots, t_n) that specifies the value of each input parameter in \mathcal{I} . The goal is to generate a smallest-sized test suite (i.e., a set of test cases) that covers all combinations of each $O \in \mathcal{O}$. The decision version of TSG is known to be NP-complete.

In this paper, we present new families of $(\mathcal{I}, \mathcal{O}, \kappa)$ for which optimal test suites can be constructed efficiently. They differ from the ones already known by the way we characterize $(\mathcal{I}, \mathcal{O})$ and κ . We then use these instances to generate test suites for arbitrary software systems. When each $O \in \mathcal{O}$ has $|O| = 2$, the sizes of the test suite are guaranteed to be at most $\lceil \log_2 n \rceil \times OPT$, matching the current best bound for this problem. Our constructions utilize the structure of $(\mathcal{I}, \mathcal{O})$ and κ ; consequently, the less “complex” $(\mathcal{I}, \mathcal{O})$ and κ are, the better are the bounds on the sizes of the test suites.

1 Introduction

Black-box testing is a form of software testing that seeks to determine if a program meets its specification from the behavioral or functional point-of-view. Typically, the total number of black-box test cases is extremely large. Testers are faced with the challenge of determining a subset of the test cases or *test suite* that is small enough so that all the chosen test cases can be executed within the current available resources, and extensive enough so that the test cases can expose most of the system’s defects.

Several methods for constructing test suites combinatorially (e.g. [3, 9, 10, 19]) can be framed in the following manner: let $\mathcal{I} = \{I_1, I_2, \dots, I_n\}$ be a set of n input parameters where each $I \in \mathcal{I}$ has $\kappa(I)$ data values. Let \mathcal{O} be a collection of subsets of \mathcal{I} where the interactions of the parameters in each subset $O \in \mathcal{O}$ are thought to affect the outcome of the system. A *test case* T for $(\mathcal{I}, \mathcal{O}, \kappa)$ is an n -tuple (t_1, t_2, \dots, t_n) that specifies each input parameter’s value; i.e., t_s is a value of I_s for $s = 1, \dots, n$. If $O = \{I_{s_1}, I_{s_2}, \dots, I_{s_r}\}$, then the test case T *covers* one combination of O : $(t_{s_1}, t_{s_2}, \dots, t_{s_r})$. The goal is to generate a smallest-sized test suite so that every combination of every $O \in \mathcal{O}$ is covered by some test

case in the test suite. We shall call such a test suite *optimal* and denote the problem as the *test suite generation problem (TSG)*.

Consider the example from [19]: a program has three input parameters A , B and C whose data values are respectively $\{1.5, 3.6\}$, $\{\text{North, South, East, West}\}$, and $\{\text{TDC, BDM}\}$. Thus, there are a total of $2 \times 4 \times 2 = 16$ test cases for the program. A popular method of software testing, first used in the AETG system [3], checks all pairwise interactions of the input parameters; i.e., $\mathcal{I} = \{A, B, C\}$ and $\mathcal{O} = \{\{A, B\}, \{A, C\}, \{B, C\}\}$. Eight test cases are sufficient as shown below.

A	1.5	3.6	1.5	3.6	1.5	3.6	1.5	3.6
B	North	North	South	South	East	East	West	West
C	TDC	BDM	BDM	TDC	TDC	BDM	BDM	TDC

On the other hand, Schroeder and Korel noted that this program has two output parameters W and Z . Suppose W is dependent on A and C , and Z is dependent on B only.¹ In their testing method, it is sufficient to cover all combinations of $\mathcal{O} = \{\{A, C\}, \{B\}\}$; four test cases are enough as shown below.

A	1.5	3.6	1.5	3.6
B	North	South	East	West
C	TDC	BDM	BDM	TDC

Solving TSG is likely going to be hard. Seroussi and Bshouty have shown that deciding if a size-4 test suite is sufficient for an arbitrary $(\mathcal{I}, \mathcal{O}, \kappa)$ is NP-complete [20]. In the special case when \mathcal{O} consists of all the t -wise combinations of \mathcal{I} for some $t \in \mathbf{Z}^+$, a test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$ is called a *mixed (t -wise) covering array*. Additionally, when κ is a constant function, i.e., all $I \in \mathcal{I}$ have the same number of data values k , the test suite is simply called a *(t -wise) covering array*. Many efficient² constructions of optimal-sized covering arrays exists (see Hartman [10] and references therein) most of which are obtained via results on orthogonal arrays, a type of combinatorial design. The existence of these constructions rely on the size of \mathcal{I} and the values of t and k . There are, however, far fewer constructions available for creating optimal mixed covering arrays. Among them is the work by Moura et al. [16] where they were able to determine the optimal size of the test suites when $|\mathcal{I}| = 4$ and for some cases when $|\mathcal{I}| = 5$.

We note though that knowing how to create optimal mixed covering arrays does not mean that we can also create optimal test suites for TSG. To see this, suppose a software system has four input parameters I_1, I_2, I_3, I_4 with 15, 2, 3, 10 data values respectively and $\mathcal{O} = \{\{I_1, I_2\}, \{I_2, I_3\}, \{I_3, I_4\}\}$. It is easy to create a test suite of 30 test cases for this software system – which is optimal since I_1 and I_2 have $15 \times 2 = 30$ combinations. If we take a mixed covering arrays approach, we will have to replace \mathcal{O} with \mathcal{O}' where the latter consists of all pairwise tuples of \mathcal{I} . Thus, although our original problem does not require it, the mixed covering array will have to make sure that all $15 \times 10 = 150$ combinations of

¹Such input-output parameter dependencies are usually found by performing some analysis on the program.

²By “efficient” here we mean that the runtimes of the constructions are polynomial in the size of the input *and* the output.

I_1 and I_4 are covered so its size is at least 150, five times more than the size of the optimal test suite.

Since TSG is NP-hard, a lot of work has also been devoted to *approximating* the best test suite. In the context of finding the smallest pairwise covering array, Cohen et al. [3] suggested the following greedy algorithm: Let \mathcal{C} consist of all the combinations of all the subsets $O \in \mathcal{O}$. At each iteration, (i) pick a test case T that covers the most number of combinations in \mathcal{C} , (ii) remove from \mathcal{C} the combinations covered by T , and (iii) add T to \mathcal{T} . Stop when \mathcal{C} is empty; i.e. all combinations of each $O \in \mathcal{O}$ have been covered. They showed that the resulting test suite \mathcal{T} is guaranteed to have size at most $1 + \log_2 N \times k^2 \leq 1 + \log_2 N \times OPT$ where k is the number of data values of all the parameters in \mathcal{I} , $N = \binom{n}{2} k^2$, the total number of combinations of all $O \in \mathcal{O}$, and OPT is the size of an optimal test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$. Cheng et al. [2] noted that, in fact, the greedy algorithm is applicable to *all* instances of TSG, and it can be shown that the size of the test suite it produces is at most $(1 + \ln |\mathcal{O}|) \times OPT$. The approximation factor in this bound is slightly better because it is dependent on the size of \mathcal{O} only and not on N . Unfortunately, while the algorithm produces small test suites, its running time is slow because finding the test case T at every iteration in the algorithm is non-trivial (e.g., see [6]). Many researchers have designed heuristics for finding T or modified the algorithm altogether to make it more efficient (e.g., [3, 12, 21]). Others [4, 5] used techniques like hill-climbing and simulated annealing to generate test suites. In empirical tests, these heuristics seem to produce small test suites but do not have provably good bounds on the sizes of the resulting test suites.

More recently, Colbourn et al. [6] found a way to speed up the above greedy algorithm for finding small mixed pairwise covering arrays. Recall that every combination of every pair of parameters $\{I_i, I_j\}$ of \mathcal{I} must be covered by the covering array. At the beginning of each iteration of the greedy algorithm, let $r_{i,j}$ be the number of combinations of $\{I_i, I_j\}$ that still lie in \mathcal{C} . Thus, $|\mathcal{C}| = \sum_{\{i,j\}} r_{i,j}$. Instead of picking a test case that covers the *most* number of combinations in \mathcal{C} at each iteration, they choose a test case T' that is guaranteed to cover at least $\delta = \sum_{\{i,j\}} \frac{r_{i,j}}{\kappa(I_i)\kappa(I_j)}$ of the combinations in \mathcal{C} . If we let I_1 and I_2 be the two parameters with the largest number of data values so that $\kappa(I_1)\kappa(I_2) \geq \kappa(I_i)\kappa(I_j)$ for every pair of parameters $I_i, I_j \in \mathcal{I}$, then $\delta \geq \sum_{\{i,j\}} \frac{r_{i,j}}{\kappa(I_1)\kappa(I_2)} \geq \frac{|\mathcal{C}|}{\kappa(I_1)\kappa(I_2)}$. That is, T' always covers at least a constant fraction of the combinations in \mathcal{C} . One can then show that the number of iterations the algorithm would take for \mathcal{C} to be empty is at most $1 + \log_2 N \times OPT$, where N is again the total number of combinations of all the pairs in \mathcal{O} . In their paper, Colbourn et al. outlined an efficient method for choosing T' . Hence, their modified greedy algorithm is a polynomial-time algorithm that produces mixed pairwise covering arrays of size at most $1 + \log_2 N \times OPT$. Note, however, that because the greedy algorithm bound of $(1 + \ln |\mathcal{O}|) \times OPT$ is applicable to all instances of TSG, we shall use it as the basis for comparison of our algorithms' performances.

Our Results. Since \mathcal{O} is a collection of subsets of \mathcal{I} , the pair $(\mathcal{I}, \mathcal{O})$ is a hypergraph. In particular, if each set in \mathcal{O} has size 2, $(\mathcal{I}, \mathcal{O})$ is a graph. We adopt this perspective throughout our work. In this paper, we present new families of software systems $(\mathcal{I}, \mathcal{O}, \kappa)$ for which we can construct optimal test suites efficiently. Our results differ from the ones known for covering arrays and mixed covering arrays in that we do not assume that \mathcal{O} contains all t -wise combinations of \mathcal{I} for some t . Our first family of software systems is

completely characterized by the structure of $(\mathcal{I}, \mathcal{O})$. In particular, we prove that if $(\mathcal{I}, \mathcal{O})$ is a bipartite graph, a cycle, or a hypertree then, for *any* function κ , $(\mathcal{I}, \mathcal{O}, \kappa)$ has an optimal test suite that can be constructed efficiently. The construction is graph theoretic in nature – a departure from the techniques used for creating optimal covering arrays. Our second family of software systems is characterized by a combination of the structure of $(\mathcal{I}, \mathcal{O})$ and the function κ , and is based on constructions of orthogonal arrays and ordered orthogonal arrays. Finally, our third family of software systems is characterized, in its simplest version, by the function κ alone. We show that as long as $\kappa(I_s)$ and $\kappa(I_t)$ are relatively prime for every pair of parameters I_s and I_t in \mathcal{I} then, for *any* $(\mathcal{I}, \mathcal{O})$, $(\mathcal{I}, \mathcal{O}, \kappa)$ has an optimal test suite that can be constructed efficiently. The construction is a direct consequence of the Chinese Remainder Theorem.

We then use our first and second families of software systems to create test suites for arbitrary software systems. When $(\mathcal{I}, \mathcal{O})$ is a graph, our strategy produces a test suite \mathcal{T} whose size is guaranteed to be at most $\lceil \log_2 n \rceil \times OPT$ in time polynomial in the size of $(\mathcal{I}, \mathcal{O})$ and \mathcal{T} . Thus, we match the bound of the greedy algorithm but improve its running time. Since our strategy is also sensitive to the structure of $(\mathcal{I}, \mathcal{O})$, the “simpler” $(\mathcal{I}, \mathcal{O})$ is, the better is the bound on the size of \mathcal{T} . We present our first two constructions in Sections 2 and 3. We present our third construction and conclude in Section 4.

We note that Meagher and Stevens [15] were the first ones to consider covering arrays for graphs. Shortly after we submitted this paper, we learned that Meagher et al. independently proved in [14] that when $(\mathcal{I}, \mathcal{O})$ is a bipartite graph or a cycle then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an optimal test suite for any κ . Their constructions have some similarities with ours.

Some definitions. Let $\mathcal{I} = \{I_1, \dots, I_n\}$ be the set of n input parameters of a software system where each $I \in \mathcal{I}$ has $\kappa(I)$ data values. If $\kappa(I) = k$, we represent the k data values of I_s as the integers in $[k] = \{0, 1, \dots, k-1\}$. Let $\mathcal{O} = \{O_1, \dots, O_m\}$ where each $O \in \mathcal{O}$ is a subset of \mathcal{I} . The software system is now described by the triple $(\mathcal{I}, \mathcal{O}, \kappa)$. A *test case* T for the software system $(\mathcal{I}, \mathcal{O}, \kappa)$ is an n -tuple (t_1, t_2, \dots, t_n) where $t_s \in [\kappa(I_s)]$ for each s . Suppose $O = \{I_{s_1}, I_{s_2}, \dots, I_{s_r}\}$. Each test case $T = (t_1, t_2, \dots, t_n)$ covers exactly one combination of O : $(t_{s_1}, t_{s_2}, \dots, t_{s_r})$. A *test suite* \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$ is a collection of test cases that covers *all* the $\kappa(I_{s_1}) \times \kappa(I_{s_2}) \times \dots \times \kappa(I_{s_r})$ combinations of O , for each $O \in \mathcal{O}$. We shall store the test cases of \mathcal{T} as columns of an $n \times |\mathcal{T}|$ array whose rows are indexed by I_1, \dots, I_n .³ If \mathcal{T} has the smallest size among all the test suites of $(\mathcal{I}, \mathcal{O}, \kappa)$, we say that \mathcal{T} is an *optimal* test suite. In particular, if \mathcal{T} has exactly $\epsilon(\mathcal{O}, \kappa) = \max\{\prod_{I_s \in O} \kappa(I_s), O \in \mathcal{O}\}$ test cases, it must be optimal.

In Section 2, we use a type of test suites called *equitable test suite* as building blocks. We say that a test suite \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$ is *equitable* if, for each $I_s \in \mathcal{I}$, every value of I_s occurs at least $\lfloor |\mathcal{T}|/\kappa(I_s) \rfloor$ times, and for values $0, 1, \dots, z$ where $z = (|\mathcal{T}| - 1) \bmod \kappa(I_s)$, exactly $\lceil |\mathcal{T}|/\kappa(I_s) \rceil$ times. It is easy to increase the size of \mathcal{T} and still keep it equitable: simply add the test case (t_1, t_2, \dots, t_n) where $t_s = |\mathcal{T}| \bmod \kappa(I_s)$ for $s = 1, \dots, n$. We call this operation $\text{ADD}(\mathcal{T})$. We will also need to decrease the number of values of a particular input parameter I_s from k_s to k'_s in \mathcal{T} . Let \mathcal{T}' denote the set of test cases in \mathcal{T} whose I_s -values are in $[k'_s]$. Let $\mathcal{T}'' = \mathcal{T} - \mathcal{T}'$. Initialize z to $|\mathcal{T}'|$. For each $T \in \mathcal{T}''$, do the following:

³Throughout the paper, we will continually be interchanging our two views of \mathcal{T} as a collection of test cases and as an array.

replace the I_s -value of T with $t_s = z \bmod \kappa(I_s)$, add the modified T to \mathcal{T}' , and increment z by 1. (Note that z keeps track of the size of \mathcal{T}' .) Call this operation $\text{MODIFY}(I_s, k_s, k'_s)$. When MODIFY changes the I_s -value of each $T \in \mathcal{T}''$, it is choosing a value of I_s which ADD would have chosen if it were to add a new test suite to \mathcal{T}' . Since ADD ensures equitability, then so does MODIFY . In section 3, we consider *strongly equitable test suites*; a test suite is *strongly equitable* if, for each $O_j = \{I_{s_1}, I_{s_2}, \dots, I_{s_r}\} \in \mathcal{O}$, every combination of O_j occurs between $\lfloor |\mathcal{T}| / \prod_{i=1}^r \kappa(I_{s_i}) \rfloor$ and $\lceil |\mathcal{T}| / \prod_{i=1}^r \kappa(I_{s_i}) \rceil$ times.

2 Construction 1: a graph-theoretic approach

In this section, we will first consider software systems $(\mathcal{I}, \mathcal{O}, \kappa)$ where $(\mathcal{I}, \mathcal{O})$ is a graph; i.e., each $O \in \mathcal{O}$ contains two parameters. We begin by describing ways in which we can derive a test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$ based on test suites of several $(\mathcal{I}', \mathcal{O}', \kappa_{|\mathcal{I}'})$ where each $(\mathcal{I}', \mathcal{O}')$ is a smaller graph compared to $(\mathcal{I}, \mathcal{O})$.

Lemma 2.1. *For $(\mathcal{I}, \mathcal{O}, \kappa)$, let $G = (\mathcal{I}, \mathcal{O})$ be a connected graph. Suppose I is a cut vertex of G and partitions the graph into $r > 1$ connected components. For $i = 1, \dots, r$, let $G_i = (\mathcal{I}_i, \mathcal{O}_i)$ denote the graph induced by the vertices of the i th component together with I . If \mathcal{T}_i is an equitable test suite for $(\mathcal{I}_i, \mathcal{O}_i, \kappa_{|\mathcal{I}_i})$, $i = 1, \dots, r$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite of size at most $M = \max(|\mathcal{T}_1|, \dots, |\mathcal{T}_r|)$.*

Proof: For each i , do the following: (i) if $|\mathcal{T}_i| < M$, increase its size to M test cases by applying ADD $M - |\mathcal{T}_i|$ times; (ii) arrange the test cases as columns of an array so that their I -values are sorted in increasing order. Once the r arrays have been constructed, concatenate them vertically. There will be r rows indexed by I all of which are identical because each \mathcal{T}_i is equitable; keep only one of the rows. Call this test suite \mathcal{T} . Clearly, \mathcal{T} has only M test cases and is equitable. Moreover, each \mathcal{T}_i covers all the combinations of \mathcal{O}_i so \mathcal{T} covers all the combinations of \mathcal{O} because $\mathcal{O} = \cup_{i=1}^r \mathcal{O}_i$. \square

Let $G = (V, E)$ be a graph, $V' \subseteq V$ and $v \in V'$. In the next lemma, we let $N(v)$ denote the neighbors of v in G . We will also *contract* V' to v (i.e., “shrink” V' to v) so that the resulting graph has $V - V' \cup \{v\}$ as its vertex set, and each edge (x, y) that has $x \in V'$ is replaced by (v, y) . Loops and multiple edges will be deleted.

Lemma 2.2. *For $(\mathcal{I}, \mathcal{O}, \kappa)$, let I_z be one of the parameters in \mathcal{I} with the most number of data values. Suppose $N(I_z)$ is an independent set and, among all the parameters in $N(I_z)$, I_y has the most number of data values. Let $(\mathcal{I}', \mathcal{O}', \kappa_{|\mathcal{I}'})$ be obtained by contracting $N(I_z)$ to I_y . If \mathcal{T}' is an equitable test suite of $(\mathcal{I}', \mathcal{O}', \kappa_{|\mathcal{I}'})$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite of size at most $|\mathcal{T}'|$.*

Proof: Arrange the test cases of \mathcal{T}' as columns of an array. Let us extend \mathcal{T}' to a test suite \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$ as follows: for each $I \in N(I_z) - \{I_y\}$, (i) append a row indexed by I that is an exact copy of the row indexed by I_y ; (ii) decrease I 's number of values from $\kappa(I_y)$ to $\kappa(I)$ using MODIFY .

The number of columns in the array was never increased so $|\mathcal{T}| = |\mathcal{T}'|$. By building \mathcal{T} from \mathcal{T}' , we guarantee that (i) \mathcal{T} is equitable because \mathcal{T}' is equitable and MODIFY preserves equitability, and (ii) if $O \in \mathcal{O} \cap \mathcal{O}'$ then \mathcal{T} covers all combinations of O because \mathcal{T}' does. If

$O \in \mathcal{O} - \mathcal{O}'$ then O must consist of a pair (I, I_t) where $I \in N(I_z) - \{I_y\}$ and $I_t \in \mathcal{I} - N(I_z)$ because $N(I_z)$ is an independent set. Thus, $(I_y, I_t) \in \mathcal{O}'$ and so \mathcal{T}' covers all combinations of (I_y, I_t) . But I 's row in \mathcal{T} is an exact copy of I_y 's row except that the values of I_y that do not belong to $[\kappa(I)]$ are modified. Hence, \mathcal{T} covers all combinations of (I, I_t) as well. \square

We are now ready for our first major result in this section.

Theorem 2.3. *If $(\mathcal{I}, \mathcal{O})$ is a bipartite graph, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

Proof: Assume for now that $G = (\mathcal{I}, \mathcal{O})$ is connected. When $|\mathcal{I}| = 2$, $\mathcal{I} = \{I_1, I_2\}$ and $\mathcal{O} = \{(I_1, I_2)\}$. A test suite consisting of all the ordered pairs of $[\kappa(I_1)] \times [\kappa(I_2)]$ is valid, equitable, and optimal for $(\mathcal{I}, \mathcal{O}, \kappa)$. Assuming the theorem is true when $|\mathcal{I}| \leq r$, let us now show it is true when $|\mathcal{I}| = r + 1$. Let I_z be the parameter with the most number of values in \mathcal{I} . Consider all its neighbors.

Case 1. I_z has only one neighbor I_y . In this case, I_y is a cut vertex in G . Let G_1 consist of the single edge (I_z, I_y) , and let $G_2 = (\mathcal{I}_2, \mathcal{O}_2)$ where $\mathcal{I}_2 = \mathcal{I} - \{I_z\}$ and $\mathcal{O}_2 = \mathcal{O} - \{(I_y, I_z)\}$. We know that $(G_1, \kappa_{|I_z, I_y})$ has an equitable test suite of size $\kappa(I_z)\kappa(I_y)$. Since G_2 remains a connected bipartite graph and \mathcal{I}_2 has r vertices, $(\mathcal{I}_2, \mathcal{O}_2, \kappa_{|\mathcal{I}_2})$ has an equitable test suite of size $\epsilon(\mathcal{O}_2, \kappa_{|\mathcal{I}_2})$ by the induction hypothesis. By Lemma 2.1, $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite of size $M = \max(\kappa(I_z)\kappa(I_y), \epsilon(\mathcal{O}_2, \kappa_{|\mathcal{I}_2}))$, which is equal to $\epsilon(\mathcal{O}, \kappa)$ since the edges of G are either in G_1 or G_2 .

Case 2: I_z has more than one neighbor. Since G is bipartite, $N(I_z)$ is an independent set. Let I_y be the parameter in $N(I_z)$ with the most number of values. Let $G' = (\mathcal{I}', \mathcal{O}')$ be derived from G by contracting $N(I_z)$ to I_y . Since $|N(I_z)| > 1$, $|\mathcal{I}'| \leq r$. Furthermore, G' remains a connected bipartite graph so by the induction hypothesis $(\mathcal{I}', \mathcal{O}', \kappa_{|\mathcal{I}'})$ has an equitable test suite of size $\epsilon(\mathcal{O}', \kappa_{|\mathcal{I}'})$. By Lemma 2.2, $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite of the same size. Let us now verify that $\epsilon(\mathcal{O}', \kappa_{|\mathcal{I}'})$ is equal to $\epsilon(\mathcal{O}, \kappa)$.

Consider all the sets that belong to \mathcal{O} but not to \mathcal{O}' , and vice versa. Let $N(N(I_z))$ consist of the neighbors of the neighbors of I_z ; i.e., $N(N(I_z)) = \cup_{I_t \in N(I_z)} N(I_t) - \{I_z\}$. The set $\mathcal{O} - \mathcal{O}'$ consists of sets of the form $\{I_t, I_z\}$ and $\{I_t, I_v\}$, where $I_t \in N(I_z)$ and $I_v \in N(N(I_z))$ while $\mathcal{O}' - \mathcal{O}$ consists of sets of the form $\{I_y, I_v\}$ where $I_v \in N(N(I_z))$. But for such I_t 's and I_v 's, $\kappa(I_y) \geq \kappa(I_t)$ and $\kappa(I_z) \geq \kappa(I_v)$ by our choice of I_y and I_z . Hence, $\kappa(I_y)\kappa(I_z) \geq \kappa(I_t)\kappa(I_z) \geq \kappa(I_t)\kappa(I_v)$, and $\kappa(I_y)\kappa(I_z) \geq \kappa(I_y)\kappa(I_v)$. It follows that $\epsilon(\mathcal{O} - \mathcal{O}', \kappa) \leq \kappa(I_y)\kappa(I_z)$ and $\epsilon(\mathcal{O}' - \mathcal{O}, \kappa_{|\mathcal{I}'}) \leq \kappa(I_y)\kappa(I_z)$. And since $\{I_y, I_z\} \in \mathcal{O} \cap \mathcal{O}'$, it must be the case that the sets with the largest number of combinations in \mathcal{O} and in \mathcal{O}' belong to $\mathcal{O} \cap \mathcal{O}'$. Thus, $\epsilon(\mathcal{O}, \kappa) = \epsilon(\mathcal{O} \cap \mathcal{O}', \kappa)$ and $\epsilon(\mathcal{O}', \kappa_{|\mathcal{I}'}) = \epsilon(\mathcal{O} \cap \mathcal{O}', \kappa)$ so $\epsilon(\mathcal{O}', \kappa_{|\mathcal{I}'}) = \epsilon(\mathcal{O}, \kappa)$.

We have now shown by induction that if $G = (\mathcal{I}, \mathcal{O})$ is a connected bipartite graph then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite of size $\epsilon(\mathcal{O}, \kappa)$. It is straightforward to check that the theorem remains true even if G is not connected. \square

In the appendix, we transform the proof above into a recursive algorithm that runs in $O(|\mathcal{I}||\mathcal{O}| + |\mathcal{I}|\epsilon(\mathcal{O}, \kappa))$. Let us now work out an example on constructing test suites for $(\mathcal{I}, \mathcal{O}, \kappa)$ when $(\mathcal{I}, \mathcal{O})$ is a bipartite graph.

Let $(\mathcal{I}, \mathcal{O}, \kappa)$ be represented by graph G_1 shown in Figure 1. The numbers next to each node indicate that parameter's number of data values. For $i = 2, \dots, 5$, G_i is obtained from

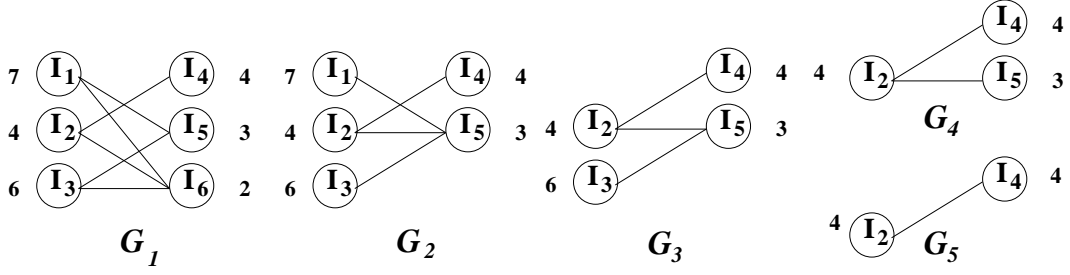


Figure 1: An example of how a bipartite graph is reduced to K_2 .

G_{i-1} by removing I_z when it has only one neighbor, and by contracting all of $N(I_z)$ into I_y otherwise. Thus, G_2 was obtained from G_1 by contracting $N(I_1)$ to I_5 , G_3 from G_2 by removing I_1 , etc. To generate a test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$, we start with a test suite \mathcal{T}_5 for G_5 .

$$\mathcal{T}_5 = \begin{bmatrix} I_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ I_4 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \end{bmatrix}$$

Since $N(I_2)$ was contracted to I_4 , the row indexed by I_5 in \mathcal{T}_4 is a copy of I_4 's where the number 3 is replaced by other numbers in $\{0, 1, 2\}$ using MODIFY.

$$\mathcal{T}_4 = \begin{bmatrix} I_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ I_4 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ I_5 & 0 & 1 & 2 & 0 & 0 & 1 & 2 & 1 & 0 & 1 & 2 & 2 & 0 & 1 & 2 & 0 \end{bmatrix}$$

The parameter I_3 was deleted from G_3 so the number of test cases in \mathcal{T}_4 is increased first so that \mathcal{T}_3 would have enough to cover the combinations of I_3 and I_5 . These test cases are marked in bold case below. Then the test cases in \mathcal{T}_4 are sorted based on their I_5 -values and the row for I_3 is appended. (Note that according to Lemma 2.1, we should append two rows first – one for I_5 and one for I_3 – so that all the combinations of (I_5, I_3) are covered and then remove the row for I_5 . Clearly, this is equivalent to just appending the row of I_3 .)

$$\mathcal{T}_3 = \begin{bmatrix} I_2 & 0 & 0 & 1 & 2 & 3 & 3 & 0 & 1 & 1 & 2 & 3 & \mathbf{0} & 0 & 1 & 2 & 2 & 3 & \mathbf{1} \\ I_4 & 0 & 3 & 0 & 0 & 0 & 3 & 1 & 1 & 3 & 1 & 1 & \mathbf{0} & 2 & 2 & 2 & 3 & 2 & \mathbf{1} \\ I_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 2 & 2 & 2 & 2 & 2 & \mathbf{2} \\ I_3 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

The parameter I_1 was deleted from G_2 so the test cases in \mathcal{T}_3 are increased again and sorted based on their I_5 values. The row for I_1 is appended so that all combinations of (I_1, I_5) are covered.

$$\mathcal{T}_2 = \begin{bmatrix} I_2 & 0 & 0 & 1 & 2 & 3 & 3 & \mathbf{2} & 0 & 1 & 1 & 2 & 3 & 0 & \mathbf{3} & 0 & 1 & 2 & 2 & 3 & 1 & \mathbf{0} \\ I_4 & 0 & 3 & 0 & 0 & 0 & 3 & \mathbf{2} & 1 & 1 & 3 & 1 & 1 & 0 & \mathbf{3} & 2 & 2 & 2 & 3 & 2 & 1 & \mathbf{0} \\ I_5 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 2 & 2 & 2 & 2 & 2 & 2 & \mathbf{2} \\ I_3 & 0 & 1 & 2 & 3 & 4 & 5 & \mathbf{0} & 0 & 1 & 2 & 3 & 4 & 5 & \mathbf{1} & 0 & 1 & 2 & 3 & 4 & 5 & \mathbf{2} \\ I_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

Finally, $N(I_1)$ was contracted into I_5 so the row indexed by I_6 in \mathcal{T}_1 is a copy of I_5 's where the number 2 is replaced by values in $\{0, 1\}$ using MODIFY.

$$\mathcal{T}_1 = \begin{bmatrix} I_2 & 0 & 0 & 1 & 2 & 3 & 3 & 2 & 0 & 1 & 1 & 2 & 3 & 0 & 3 & 0 & 1 & 2 & 2 & 3 & 1 & 0 \\ I_4 & 0 & 3 & 0 & 0 & 0 & 3 & 2 & 1 & 1 & 3 & 1 & 1 & 0 & 3 & 2 & 2 & 2 & 3 & 2 & 1 & 0 \\ I_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ I_3 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 0 & 1 & 2 & 3 & 4 & 5 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 2 \\ I_1 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ I_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

We also use the same approach in the proof of Theorem 2.3 to show that the next theorem is true.

Theorem 2.4. *If $(\mathcal{I}, \mathcal{O})$ is a cycle, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

Proof: Since an even cycle is bipartite, we only have to consider the case when $(\mathcal{I}, \mathcal{O})$ is an odd cycle. Once again, we will prove the theorem by induction. When $|\mathcal{I}| = 3$, it is easy to construct an equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$ for $(\mathcal{I}, \mathcal{O}, \kappa)$. Assuming the theorem holds when $(\mathcal{I}, \mathcal{O})$ is an odd cycle with at most $2r - 1$ nodes, let us now consider the case when $|\mathcal{I}| = 2r + 1$. Let I_z the parameter with the most number of values in \mathcal{I} . Notice that I_z has two non-adjacent neighbors, say I_y and I_x ; let $\kappa(I_y) \geq \kappa(I_x)$. Contract $N(I_z)$ to I_y . The resulting graph $G' = (\mathcal{I}', \mathcal{O}')$ consists of a smaller odd cycle $(\mathcal{I}'', \mathcal{O}'')$ containing I_y together with the edge (I_y, I_z) hanging off of the cycle. By assumption, we know that $(\mathcal{I}'', \mathcal{O}'', \kappa_{|\mathcal{I}''})$ has an equitable optimal test suite of size $\epsilon(\mathcal{O}'', \kappa_{|\mathcal{I}''})$. Combining this with the fact that I_y is a cut vertex in G' , Lemma 2.1 implies that $(\mathcal{I}', \mathcal{O}', \kappa_{|\mathcal{I}'})$ has an equitable test suite of size at most $\max(\epsilon(\mathcal{O}'', \kappa_{|\mathcal{I}''}), \kappa(I_y)\kappa(I_z)) = \epsilon(\mathcal{O}', \kappa_{|\mathcal{I}'})$. But $\epsilon(\mathcal{O}', \kappa_{|\mathcal{I}'}) = \epsilon(\mathcal{O}, \kappa)$ because every edge (I_s, I_t) in $(\mathcal{I}', \mathcal{O}')$ is also in $(\mathcal{I}, \mathcal{O})$ or $\kappa(I_s)\kappa(I_t) \leq \kappa(I_y)\kappa(I_z)$. By induction, we have now shown that the theorem is true. \square

We now use Theorem 2.3 to generate a test suite for software systems where $(\mathcal{I}, \mathcal{O})$ is an arbitrary graph.

Corollary 2.5. *If $(\mathcal{I}, \mathcal{O})$ is a q -colorable graph, $q \geq 2$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a test suite of size at most $\lceil \log_2 q \rceil \times \epsilon(\mathcal{O}, \kappa)$. Furthermore, provided a q -coloring of $(\mathcal{I}, \mathcal{O})$ is given, the test suite can be constructed in $O(\log q \times (|\mathcal{I}||\mathcal{O}| + |\mathcal{I}|\epsilon(\mathcal{O}, \kappa)))$ time.*

Proof: Let us show that every q -colorable graph can be covered by $\lceil \log_2 q \rceil$ of its bipartite subgraphs. First, let K_q be the complete graph on $V = \{0, 1, \dots, q - 1\}$. Dumitrescu [7] gave the following construction. For each $j \in V$, let j_i denote the i th digit in the binary representation of j . For $i = 1, \dots, \lceil \log_2 q \rceil$, let H_i be the subgraph of K_q whose edge set consists of pairs (j, j') such that $j_i \neq j'_i$. H_i is bipartite because V can be partitioned into two sets: one contains all j such that $j_i = 0$ and another contains all j such that $j_i = 1$. Since any two integers in V differ in at least one digit in their binary representation, every edge in K_q is covered by at least one graph in $\{H_1, \dots, H_{\lceil \log_2 q \rceil}\}$.

Next, consider a q -coloring of $G = (\mathcal{I}, \mathcal{O})$ using the colors $0, 1, \dots, q - 1$. Let \mathcal{I}_j denote the set of parameters in G assigned the color j . For $i = 0, 1, \dots, \lceil \log_2 q \rceil$, let G_i be the subgraph of G whose edge set consists of edges between \mathcal{I}_j and $\mathcal{I}_{j'}$ provided $j_i \neq j'_i$. By the

same argument as above, the graphs in $\{G_1, \dots, G_{\lceil \log_2 q \rceil}\}$ are bipartite and cover all the edges of G . It is easy to check that once the q -coloring of G is found, generating the G_i 's take $O(|\mathcal{O}| \log q)$ time.

Let $(\mathcal{I}, \mathcal{O}_i, \kappa)$ denote the software system that corresponds to G_i . From Theorem 2.3, $(\mathcal{I}, \mathcal{O}_i, \kappa)$ has an equitable test suite \mathcal{T}_i with size $\epsilon(\mathcal{O}_i, \kappa) \leq \epsilon(\mathcal{O}, \kappa)$. By the construction of the G_i 's, it follows that $\mathcal{T} = \bigcup_{i=1}^{\lceil \log_2 q \rceil} \mathcal{T}_i$ covers all combinations of all the sets in \mathcal{O} , and has size $\sum_{i=1}^{\lceil \log_2 q \rceil} \epsilon(\mathcal{O}_i, \kappa) \leq \lceil \log_2 q \rceil \times \epsilon(\mathcal{O}, \kappa)$. Finally, because we have to construct $\lceil \log_2 q \rceil$ different test suites, and we know that each test suite can be constructed in $O(|\mathcal{I}||\mathcal{O}| + |\mathcal{I}|\epsilon(\mathcal{O}, \kappa))$ time, the runtime in the corollary follows. \square

Let n^* be the number of nodes in the largest connected component of $(\mathcal{I}, \mathcal{O})$. Since an n^* -coloring of $(\mathcal{I}, \mathcal{O})$ can always be found quickly, the above result implies that we can efficiently construct a test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$ whose size is at most $\lceil \log_2 n^* \rceil \times \epsilon(\mathcal{O}, \kappa)$. But notice that $|\mathcal{O}| \geq n^* - 1$ so our bound is as competitive as the one for the greedy algorithm. It is in fact better if $(\mathcal{I}, \mathcal{O})$ is a dense graph, or q is smaller than n^* and a q -coloring of $(\mathcal{I}, \mathcal{O})$ can be found quickly. In the next section, we shall show that our bound can be further improved when the function κ satisfies certain conditions.

Next, we extend our technique to hypertrees for our second main result in this section. Given a hypergraph H , a sequence $v_1 e_1 v_2 e_2 \dots v_p e_p v_{p+1}$ is a *path* if the v_i 's and e_i 's are distinct vertices and edges in H , and $v_i, v_{i+1} \in e_i$ for $i = 1, \dots, p$. If $v_{p+1} = v_1$, the sequence is a *cycle*. It is easy to verify that in a cycle-free hypergraph, any two edges have at most one vertex in common. A hypergraph is *connected* if, for every pair of vertices v_i and v_j , there is a path from v_i to v_j . It is a *hypertree* if it is cycle-free and connected.

Lemma 2.6. *Let H be a hypertree and e be an edge in H . Deleting e from H creates $|e|$ connected components each of which is also a hypertree.*

Proof: Let H' denote the hypergraph obtained by deleting e from H . Since H is acyclic, every connected component of H' is acyclic as well (i.e., it is a hypertree). Furthermore, every $v_i \in e$ must belong to some connected component of H' (even though the component may have no edges). Since v_i cannot belong to two or more connected components of H' , H' has at most $|e|$ connected components. Now, if two vertices in e , v_i and v_j , belong to the same connected component of H' , then the path from v_i to v_j in H' together with e forms a cycle in H . This is a contradiction. Hence, H' has exactly $|e|$ connected components. \square

Theorem 2.7. *When $(\mathcal{I}, \mathcal{O})$ is a hypertree, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite with size $\epsilon(\mathcal{O}, \kappa)$.*

Proof: If $(\mathcal{I}, \mathcal{O})$ contains only one edge O , the theorem is clearly true. Let us now assume that the theorem holds true as well for all hypertrees with at most r edges, and $(\mathcal{I}, \mathcal{O})$ has $r + 1$ edges. Suppose O_1 has the most number of combinations among all the sets in \mathcal{O} . Let $\mathcal{O}' = \mathcal{O} - \{O_1\}$. According to Lemma 2.6, $(\mathcal{I}, \mathcal{O}')$ is still cycle-free and has $|O_1|$ connected components all of which have at most r edges. Let us denote the i th component of $(\mathcal{I}, \mathcal{O}')$ by $H_i = (\mathcal{I}_i, \mathcal{O}_i, \kappa)$ where H_i contains the i th input parameter I_{s_i} of O_1 .

If H_i has no edges (i.e., \mathcal{I}_i is a singleton and \mathcal{O}_i is an empty set), let \mathcal{T}_i simply consist of a single row where its j th entry equals $j \bmod \kappa(I_i)$ for $j = 0, \dots, \epsilon(\mathcal{O}, \kappa) - 1$. If H_i has at least one edge, by our assumption, it has an equitable test suite \mathcal{T}_i of size $\epsilon(\mathcal{O}_i, \kappa)$. Increase

the size of \mathcal{T}_i to $\epsilon(\mathcal{O}, \kappa)$ using ADD. Let us now assemble the test cases in $\bigcup_{i=1}^{|\mathcal{O}_1|} \mathcal{T}_i$ to form a test suite \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$. The rows in \mathcal{T} will be indexed first by the parameters in H_1 , followed by the parameters in H_2 , and so forth. For each combination $(t_{s_1}, t_{s_2}, \dots, t_{s_{|\mathcal{O}_1|}})$ of \mathcal{O}_1 , do the following: (i) for $i = 1$ to $|\mathcal{O}_1|$, find an unused test case $T_i \in \mathcal{T}_i$ whose I_{s_i} -value equals t_{s_i} and mark T_i as used. Since \mathcal{T}_i is equitable, $|\mathcal{T}_i| = \epsilon(\mathcal{O}, \kappa)$, and \mathcal{O}_1 has the most number of combinations, we will always find such a T_i . (ii) Form the test case T by concatenating $T_1, T_2, \dots, T_{|\mathcal{O}_1|}$ into a single column, and add T to \mathcal{T} .

Our assembly process ensures that \mathcal{T} covers each combination of \mathcal{O}_1 . Furthermore, because the \mathcal{T}_i 's are equitable test suites that cover all the combinations of all the sets in \mathcal{O}' , \mathcal{T} is valid and equitable for $(\mathcal{I}, \mathcal{O}, \kappa)$. And since there are exactly $\epsilon(\mathcal{O}, \kappa)$ combinations of \mathcal{O}_1 , $|\mathcal{T}| = \epsilon(\mathcal{O}, \kappa)$. By induction, it follows that our theorem is true. \square

Like the test suite generation scheme in Theorem 2.3, the proof above can be transformed into a recursive algorithm that runs in time polynomial in the size of $(\mathcal{I}, \mathcal{O})$ and $\epsilon(\mathcal{O}, \kappa)$. The following corollary is immediate.

Corollary 2.8. *If the hypergraph $(\mathcal{I}, \mathcal{O})$ can be covered by q of its partial hypergraphs $(\mathcal{I}_i, \mathcal{O}_i)$, $i = 1, \dots, q$, all of which are hypertrees, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a test suite of size $q \times OPT$, where OPT is the size of an optimal test suite of $(\mathcal{I}, \mathcal{O}, \kappa)$.*

3 Construction 2: an OA and OOA-approach

In this section, we make use of a well-studied family of combinatorial objects known as *orthogonal arrays* (OA) and its generalization *ordered orthogonal arrays* (OOA) to construct our test suites. Since the introduction of covering arrays, orthogonal arrays have been used to create test suites. We are presenting Theorem 3.1 and its corollaries so that we can frame the results of orthogonal arrays in the context of our view that $(\mathcal{I}, \mathcal{O})$ is a hypergraph, and use their proofs as a gentle introduction to the rest of the results in this section.

An $n \times \lambda k^t$ array A with entries from a k -element set S is an *orthogonal array with k levels, strength t and index λ* , or $OA_\lambda(n, k, t)$ for short, if every $t \times \lambda k^t$ subarray of A contains each t -tuple of S exactly λ times as a column. Of interest to us are orthogonal arrays of index 1.

Recall that a hypergraph $H = (V, E)$ is *q -partite* if V can be partitioned into q sets so that for each $e \in E$ no two nodes in e belong to the same partite set.

Theorem 3.1. *Suppose $(\mathcal{I}, \mathcal{O})$ is a q -partite hypergraph, κ is a constant function where $\kappa(I) = k$ for each $I \in \mathcal{I}$, and $\epsilon(\mathcal{O}, \kappa) = k^t$ for some $t \in \mathbf{Z}^+$, $t \leq q$. If an $OA_1(q, k, t)$ exists then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a strongly equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

Proof: Let $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_q$ be the q partitions of \mathcal{I} . Let A be a $q \times k^t$ orthogonal array $OA_1(q, k, t)$ whose entries belong to the set $[k]$. We create a test suite \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$ by letting the row indexed by each $I \in \mathcal{I}_i$ be an exact copy of the i th row of A for $i = 1, \dots, q$. Since A has k^t columns, \mathcal{T} has k^t test cases.

Now, let $O = \{I_{s_1}, I_{s_2}, \dots, I_{s_r}\} \in \mathcal{O}$. Clearly, $r \leq t$ because $\epsilon(\mathcal{O}, \kappa) = k^t$. Furthermore, no two parameters in O belong to the same partite set so these parameters' rows in \mathcal{T} correspond to different rows in A . If $r = t$, then the subarray of \mathcal{T} formed by the rows

of $I_{s_1}, I_{s_2}, \dots, I_{s_r}$ contains each tuple of $[k]^t$ as a column exactly once because A is an orthogonal array. If $r < t$, there must be a set O' such that $O \subseteq O'$ and no two parameters in O' belong to the same partite set. Applying the previous argument, all the combinations of O' must be covered exactly once by \mathcal{T} . This means that all combinations of O are covered exactly $|\mathcal{T}|/k^{t-r} = k^r$ times. Hence, \mathcal{T} is a strongly equitable test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$. \square

Bush [1] proved that whenever k is a prime power, and $0 \leq t \leq k+1$, an $OA_1(k+1, k, t)$ exists. Thus, we have the following corollary.

Corollary 3.2. *Let k be a prime power, t be an integer such that $0 \leq t \leq k+1$. If $(\mathcal{I}, \mathcal{O})$ is a $(k+1)$ -partite hypergraph, κ is a constant function with $\kappa(I) = k$ for each $I \in \mathcal{I}$, and $\epsilon(\mathcal{O}, \kappa) = k^t$, then $(\mathcal{I}, \mathcal{O}, k)$ has a strongly equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

The above result can now be applied to the special case when $(\mathcal{I}, \mathcal{O})$ is a graph.

Corollary 3.3. *Let k be a prime power. If $(\mathcal{I}, \mathcal{O})$ is a q -colorable graph, $q \geq 2$, and $\kappa(I) = k$ for each $I \in \mathcal{I}$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a test suite of size at most $\lceil \log_{k+1} q \rceil \times \epsilon(\mathcal{O}, \kappa)$.*

Proof: When $(\mathcal{I}, \mathcal{O})$ is a q -partite graph, we employ the same technique in Corollary 2.5 (except that we use the $(k+1)$ -ary representation of an integer rather than its binary representation) to create $\lceil \log_{k+1} q \rceil$ $(k+1)$ -partite subgraphs of $(\mathcal{I}, \mathcal{O})$ so that together the subgraphs cover $(\mathcal{I}, \mathcal{O})$. We then construct optimal test suites for the software systems that correspond to the $(k+1)$ -partite subgraphs using orthogonal array $OA_1(k+1, k, 2)$ on the set $[k]$, and combine them together to form a test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$. Since there are $\lceil \log_{k+1} q \rceil$ such test suites all of size k^2 , the theorem follows. \square

While the above corollary provides a better bound for the size of a test suite of $(\mathcal{I}, \mathcal{O}, \kappa)$ when κ is a constant function than the one given in Corollary 2.5, it does not seem to be particularly useful in other situations. When κ is not a constant function, simply replacing each $\kappa(I)$ with the smallest prime power k such that $k \geq \{\max \kappa(I) : I \in \mathcal{I}\}$ is not a good strategy because the size of the optimal test suite for the modified software system can be $\theta(k)$ times that of the optimal test suite of $(\mathcal{I}, \mathcal{O}, \kappa)$ when $(\mathcal{I}, \mathcal{O})$ is a graph. For example, suppose $(\mathcal{I}, \mathcal{O})$ is a 3-cycle where $\kappa(I_1) = p$, a prime number, and $\kappa(I_2) = \kappa(I_3) = 2$. The optimal test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$ has size $2p$ but rounding $\kappa(I_2)$ and $\kappa(I_3)$ to p will require a test suite of size p^2 .⁴ To fix this situation, we introduce *base- k* functions, and modify Bush's construction for orthogonal arrays so they become suitable for these types of functions.

Let k be a positive integer. We say that a function $\kappa : \mathcal{I} \rightarrow \mathbf{Z}^+$ is *base- k* if it maps each $I \in \mathcal{I}$ to a power of k . It turns out that base- k functions approximate arbitrary κ 's well in the following sense.

Lemma 3.4. *Let k be a positive integer and $(\mathcal{I}, \mathcal{O})$ be a graph. For $(\mathcal{I}, \mathcal{O}, \kappa)$, let $\kappa'(I) = k^{\lceil \log_k \kappa(I) \rceil}$. If $(\mathcal{I}, \mathcal{O}, \kappa')$ has an optimal test suite of size $\epsilon(\mathcal{O}, \kappa')$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a test suite of size at most $k^2 \times \epsilon(\mathcal{O}, \kappa)$.*

Proof: Since $\kappa(I) \leq \kappa'(I) \leq k \times \kappa(I)$ for each $I \in \mathcal{I}$, every test suite \mathcal{T}' for $(\mathcal{I}, \mathcal{O}, \kappa')$ can be transformed into a test suite \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$. Furthermore, because each $O \in \mathcal{O}$ contains two parameters, if $|\mathcal{T}'| = \epsilon(\mathcal{O}, \kappa')$ then $|\mathcal{T}| = \epsilon(\mathcal{O}, \kappa) \leq k^2 \times \epsilon(\mathcal{O}, \kappa)$. \square

⁴In this simple example, one might do some post-processing of the test suite to push the size from p^2 down to $2p$. In bigger examples, however, it is not obvious that such post-processing will eliminate many test cases.

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 1 & 0 & 3 & 2 & 2 & 3 & 0 & 1 & 3 & 2 & 1 & 0 \\ 0 & 1 & 2 & 3 & 2 & 3 & 0 & 1 & 3 & 2 & 1 & 0 & 1 & 0 & 3 & 2 \\ 0 & 1 & 2 & 3 & 3 & 2 & 1 & 0 & 1 & 0 & 3 & 2 & 2 & 3 & 0 & 1 \end{bmatrix}$$

Figure 2: An orthogonal array $OA_1(4,4,2)$ that obeys the conditions of Lemma 3.5. The elements of $GF(4)$ are represented by their indices.

Next, we describe a very simple construction of $OA_1(k,k,2)$ due to Bush [1] that is based on the Galois Field $GF(k)$ where k is a prime power. Let A be a $k \times k^2$ array. Index its rows by the elements of $GF(k)$: $\alpha_0, \dots, \alpha_{k-1}$, and its columns by the k^2 polynomials of maximum degree 1 over $GF(k)$: $\phi_0, \dots, \phi_{k^2-1}$. Set the value of A_{ij} to $\phi_j(\alpha_i)$ for each i and j . Now consider a $2 \times k^2$ subarray of A . No two of its columns indexed by say, ϕ_j and $\phi_{j'}$, are equal because $\phi_j - \phi_{j'}$ is a polynomial of maximum degree at most 1 and cannot have more than one zero. Hence, all the pairwise combinations of the elements of $GF(k)$ occur exactly once in this $2 \times k^2$ subarray of A . If the addition and multiplication tables of $GF(k)$ are known, constructing A takes $O(k^3)$ time. It turns out that the elements of $GF(k)$ can be chosen and the columns of A rearranged to obtain stronger properties about A as stated in the next lemma. The proof can be found in the appendix.

Lemma 3.5. *Let $k = p^z$ where p is a prime number and z is a positive integer. Let A be an $OA_1(k,k,2)$ orthogonal array obtained by Bush's construction. Let $A[r]$ denote the array obtained by replacing each α_i by $\alpha_{i \bmod r}$ in the entries of A . The elements of $GF(k)$ can be chosen and the columns of A can be arranged so that for each non-negative integer $z' \leq z$, every row of $A[p^{z'}]$ is a concatenation of permutations of $\{\alpha_0, \alpha_1, \dots, \alpha_{p^{z'}-1}\}$.*

See an example of an orthogonal array that satisfies the conditions of Lemma 3.5 in Figure 2. We are now ready for the first main result of this section.

Theorem 3.6. *Let k be a prime power and $z \in \mathbf{Z}^+$. If $(\mathcal{I}, \mathcal{O})$ is a k^z -partite graph, κ is a base- k function, and $\epsilon(\mathcal{O}, \kappa) = k^{2z}$ or k^{2z+1} , then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a strongly equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

Proof: Let us first assume that $\epsilon(\mathcal{O}, \kappa) = k^{2z}$. Let the k^z partite sets of \mathcal{I} be $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_{k^z}$. Let A be a $k^z \times k^{2z}$ orthogonal array $OA_1(k^z, k^z, 2)$ obtained using Bush's construction and an array that respects the properties in Lemma 3.5. Recall that $A[r]$ is the array obtained by replacing each α_i by $\alpha_{i \bmod r}$ in the entries of A . Replace each α_i in $A[r]$ by i so that their entries belong to $[r]$ for $r = 1, k, k^2, \dots, k^z$. We will construct a test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$ in a manner very similar to the one we used in Theorem 3.1 except that the rows of \mathcal{T} will be copied from $A[k^z], A[k^{z-1}], \dots, A[k^0]$ and not just A .

Since κ is a base- k function and $\epsilon(\mathcal{O}, \kappa) = k^{2z}$, there are at most $2z + 1$ distinct number of data values among the parameters in $\mathcal{I} : \{k^0, k^1, k^2, \dots, k^{2z}\}$. For each partite set \mathcal{I}_j , let $I_{j,l}$ denote a parameter in \mathcal{I}_j with k^l values.

- If $l \leq z$, let the row indexed by $I_{j,l}$ in \mathcal{T} be an exact copy of the j th row in $A[k^l]$.
- If $l > z$, let the row indexed by $I_{j,l}$ in \mathcal{T} consist of k^{2z-l} 0's, k^{2z-l} 1's, \dots , k^{2z-l} $k^l - 1$'s.

Clearly, each row in \mathcal{T} has length k^{2z} so \mathcal{T} has k^{2z} test cases. Let us now prove that it is a strongly equitable test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$. Suppose $O = \{I_{j_1, l_1}, I_{j_2, l_2}\}$. Since $\epsilon(\mathcal{O}, \kappa) = k^{2z}$, it follows that $k^{l_1} k^{l_2} \leq k^{2z}$.

Case 1: $l_1 \leq z, l_2 \leq z$. When $l_1 = l_2 = z$, both the rows indexed by I_{j_1, l_1} and I_{j_2, l_2} in \mathcal{T} belong to array A and so must cover the $k^z \times k^z$ combinations of O exactly once. When $l_1, l_2 \leq z$, then for $i = 1, 2$, the row indexed by I_{j_i, l_i} is almost an exact copy of the row indexed by $I_{j_i, z}$ except that each value v is replaced by $v \bmod k^{l_i}$. We just showed that \mathcal{T} would have covered all combinations of $(I_{j_1, z}, I_{j_2, z})$ so it must do likewise for $(I_{j_1, l_1}, I_{j_2, l_2})$. Furthermore, each combination of $(I_{j_1, l_1}, I_{j_2, l_2})$ appears $k^{z-l_1} \times k^{z-l_2}$ times because of the way I_{j_i, l_i} is obtained from $I_{j_i, z}$ for $i = 1, 2$.

Case 2: $l_1 < z < l_2$ or $l_2 < z < l_1$. Without loss of generality, assume $l_1 < z < l_2$ so the row indexed by I_{j_1, l_1} is a row from $A[k^{l_1}]$. According to Lemma 3.5, and the fact that we replaced each α_i by i , the row is a sequence of k^{2z-l_1} permutations of $0, 1, \dots, k^{l_1} - 1$. On the other hand, the row indexed by I_{j_2, l_2} consists of k^{2z-l_2} 0's, k^{2z-l_2} 1's and ending with $k^{2z-l_2} k^{l_2} - 1$'s. Since k^{l_1} divides k^{2z-l_2} , it follows that every combination of O is covered exactly $k^{2z-(l_1+l_2)}$ times in \mathcal{T} .

Since $l_1, l_2 > z$ is an impossibility, we have now shown that for every $O \in \mathcal{O}$, every combination in O is covered the same number of times in \mathcal{T} . When $\epsilon(\mathcal{O}, \kappa) = k^{2z+1}$, our construction for \mathcal{T} is similar with a few modifications:

- If $l \leq z$, let the row indexed by $I_{j, l}$ in \mathcal{T} consist of k copies of the j th row in $A[k^l]$ concatenated together.
- If $l > z$, let the row consist of k^{2z+1-l} 0's, k^{2z+1-l} 1's, \dots , $k^{2z+1-l} k^l - 1$'s.

The resulting test suite \mathcal{T} is strongly equitable in this case for the same reasons when $\epsilon(\mathcal{O}, \kappa) = k^{2z}$. \square

In the above construction, creating the array A takes $O(f(k^z) + k^{3z}) = O(f(k^z) + \epsilon(\mathcal{O}, \kappa)^{1.5})$ time, where $f(k^z)$ is the time needed to construct the addition and multiplication tables of $GF(k^z)$. Filling up the entries of \mathcal{T} simply requires copying from the entries of the $A[r]$'s and so takes $O(|\mathcal{I}| \epsilon(\mathcal{O}, \kappa))$ time. Thus, if the k^z -coloring of $(\mathcal{I}, \mathcal{O})$ is given, then the test suite can be constructed in $O(f(k^z) + \epsilon(\mathcal{O}, \kappa)^{1.5} + |\mathcal{I}| \epsilon(\mathcal{O}, \kappa))$ time.

Consider the software system example in the previous section, shown in Figure 1, except that this time we replace each $\kappa(I)$ with $\kappa'(I) = 2^{\lceil \log \kappa(I) \rceil}$. Hence, $\epsilon(\mathcal{O}, \kappa) = 32 = 2^5$. According to the proof of Theorem 3.6, because $32 = 2^{2 \cdot 2 + 1}$ and $(\mathcal{I}, \mathcal{O})$ is bipartite, we can construct the rows of I_2 and I_4, I_5, I_6 based on the first two rows of the orthogonal array $OA_1(4, 4, 2)$ in Figure 2 while I_1 and I_3 's rows simply consist of 2^2 0's, 2^2 1's, and so forth ending with 2^2 7's. The test suite is shown below.

$$\begin{bmatrix} I_1 & 0000 & 1111 & 2222 & 3333 & 4444 & 5555 & 6666 & 7777 \\ I_2 & 0123 & 0123 & 0123 & 0123 & 0123 & 0123 & 0123 & 0123 \\ I_3 & 0000 & 1111 & 2222 & 3333 & 4444 & 5555 & 6666 & 7777 \\ I_4 & 0123 & 1032 & 2301 & 3210 & 0123 & 1032 & 2301 & 3210 \\ I_5 & 0123 & 1032 & 2301 & 3210 & 0123 & 1032 & 2301 & 3210 \\ I_6 & 0101 & 1010 & 0101 & 1010 & 0101 & 1010 & 0101 & 1010 \end{bmatrix}$$

Using the subgraph covering technique we used in Corollary 2.5, the next corollary is immediate.

Corollary 3.7. *Let k be a prime power and $z \in \mathbf{Z}^+$. If $(\mathcal{I}, \mathcal{O})$ is a q -colorable graph, $q \geq 2$, κ is a base- k function, and $\epsilon(\mathcal{O}, \kappa) = k^{2z}$ or k^{2z+1} then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a test suite of size at most $\lceil \log_{kz} q \rceil \times \epsilon(\mathcal{O}, \kappa)$.*

When κ is not a base- k software system, we define $\kappa'(I) = 2^{\lceil \log_2 \kappa(I) \rceil}$ and generate an optimal test suite for $(\mathcal{I}, \mathcal{O}, \kappa')$. Applying Lemma 3.4, we have the following corollary.

Corollary 3.8. *Suppose κ is not a base-2 function in $(\mathcal{I}, \mathcal{O}, \kappa)$. Let $\kappa'(I) = 2^{\lceil \log_2 \kappa(I) \rceil}$, and let z be the smallest integer so that $\epsilon(\mathcal{O}, \kappa') \leq 2^{2z}$ or 2^{2z+1} . If $(\mathcal{I}, \mathcal{O})$ is a q -colorable graph, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a test suite of size at most $4 \lceil \log_{2z} q \rceil \times \epsilon(\mathcal{O}, \kappa)$. Moreover, if the q -coloring of $(\mathcal{I}, \mathcal{O})$ is given, then the test suite can be constructed in $O(\log q \times (|\mathcal{O}| + f(2^z) + \epsilon(\mathcal{O}, \kappa)^{1.5} + |\mathcal{I}| \epsilon(\mathcal{O}, \kappa)))$ time where $f(2^z)$ is the time needed to construct the addition and multiplication tables of $GF(2^z)$.*

Hence, when $z \gg 4$ the above corollary produces a bound on the size of $(\mathcal{I}, \mathcal{O}, \kappa)$'s test suite that is better than the one found in Corollary 2.5. Next, we attempt to extend the result in Theorem 3.6 to the case when $(\mathcal{I}, \mathcal{O})$ is a general hypergraph for the second main result of this section. To do so, we make use of a generalization of orthogonal arrays.

An $nl \times \lambda k^t$ array A with entries from a k -element set S is an *ordered orthogonal array* $OOA_\lambda(n, l, k, t)$ if the rows can be partitioned into n groups of l rows each, denoted by R_1, R_2, \dots, R_n where $R_i = \{r_{ij} : 1 \leq j \leq l\}$, so that the following condition is true: whenever $t = \sum_{j=1}^n t_j$ where each t_j is a non-negative integer such that $t_j \leq l$ then the $t \times \lambda k^t$ subarray of A formed by taking the first t_j rows of R_i , $i = 1, \dots, n$, contains each t -tuple of S exactly λ times as a column. Notice that when $l = 1$, then an $OOA_\lambda(n, l, k, t)$ is an $OA_\lambda(n, k, t)$. Below is an example of an $OOA_1(2, 2, 3, 2)$.

$$\begin{bmatrix} r_{11} & 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ r_{12} & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ r_{21} & 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\ r_{22} & 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{bmatrix}$$

Theorem 3.9. *Suppose $(\mathcal{I}, \mathcal{O})$ is a q -partite hypergraph, κ is a base- k function, and $\epsilon(\mathcal{O}, \kappa) = k^t$ for some $t \in \mathbf{Z}^+$. If an $OOA_1(q, t, k, t)$ exists then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a strongly equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

Proof: Let $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_q$ be the q partite sets of \mathcal{I} . While each \mathcal{I}_i can contain many parameters, these parameters can only have $t + 1$ types of number of data values: k^0, k^1, \dots, k^t because $\epsilon(\mathcal{O}, \kappa) = k^t$. Denote by $I_{ij} \in \mathcal{I}_i$ a parameter with $\kappa(I_{ij}) = k^j$. Let A be an $OOA_1(q, t, k, t)$ whose elements are from the set $[k]$. We shall denote A 's rows as $r_{11}, \dots, r_{1t}, \dots, r_{q1}, \dots, r_{qt}$ as stated in the definition of OOA 's. We create a test suite \mathcal{T} for $(\mathcal{I}, \mathcal{O}, \kappa)$ in the following manner. For each i ,

- when $j = 0$, let the row indexed by I_{ij} simply consist of all 0's;
- when $j > 0$, let the row indexed by I_{ij} be $r_{i1}k^0 + r_{i2}k^1 + \dots + r_{ij}k^{j-1}$. That is, the h th entry in the row indexed by I_{ij} is $r_{i1}^{(h)}k^0 + r_{i2}^{(h)}k^1 + \dots + r_{ij}^{(h)}k^{j-1}$ where $r_{i*}^{(h)}$ is the h th entry in row r_{i*} of array A .

Let us now show that all the combinations of each $O \in \mathcal{O}$ are covered the same number of times. Since no two parameters in O belong to the same partite set, we can write O as $\{I_{q_1 s_1}, I_{q_2 s_2}, \dots, I_{q_l s_l}\}$.

Case 1: Each $s_i \geq 1$ and $s_1 + s_2 + \dots + s_l = t$.

In this case, O has $k^{s_1} \times k^{s_2} \times \dots \times k^{s_l} = k^t$ combinations. Let (v_1, v_2, \dots, v_l) be a particular combination of O . Each $v_i \in [k^{s_i}]$ so we can write v_i as $x_{q_i 1} k^0 + x_{q_i 2} k^1 + \dots + x_{q_i s_i} k^{s_i - 1}$ where each $x_{q_i *}$ $\in [k]$. In particular, $(x_{q_1 1}, \dots, x_{q_1 s_1}, x_{q_2 1}, \dots, x_{q_2 s_2}, \dots, x_{q_l 1}, \dots, x_{q_l s_l})$ is a tuple of $[k]^t$. But notice that this combination appears exactly once as a column in the subarray of A formed by the rows $r_{q_1 1}, \dots, r_{q_1 s_1}, r_{q_2 1}, \dots, r_{q_2 s_2}, \dots, r_{q_l 1}, \dots, r_{q_l s_l}$ because A is an $OOA_1(q, t, k, t)$. Consequently, this means that (v_1, v_2, \dots, v_l) appears exactly once as a column in the subarray of \mathcal{T} indexed by $I_{q_1 s_1}, I_{q_2 s_2}, \dots, I_{q_l s_l}$.

Case 2: Each $s_i \geq 1$ but $s_1 + s_2 + \dots + s_l < t$.

Choose s'_1 such that $s'_1 + s_2 + \dots + s_l = t$. By case 1, all the combinations of $O' = \{I_{q_1 s'_1}, I_{q_2 s_2}, \dots, I_{q_l s_l}\}$ are covered exactly once. By our construction of the rows of \mathcal{T} , the h th entry in the row indexed by $I_{q_1 s'_1}$ is the h th entry in the row indexed by $I_{q_1 s'_1} \bmod k^{s_1}$. Thus, not only should every combination of O be covered by \mathcal{T} ; each one is covered the same number of times.

Case 3: $s_1 + s_2 + \dots + s_l \leq t$ but some s_i 's are equal to 0.

In this case, choose $O' \subset O$ such that all the parameters in O' have at least k data values. By cases 1 and 2, the combinations of O' are covered the same number of times. All the parameters in $O - O'$ have their values set to 0. So all the combinations of O are also covered the same number of times. \square

Schmid [18] and Lawrence [11] independently proved that OOA 's are equivalent to (t, m, s) -nets, which are collections of points in the s -dimensional cube that have desirable uniformity properties. Many constructions for (t, m, s) -nets are known; see [13, 17] for a survey. One result due to Faure [8], which when translated in terms of OOA 's, states that if $k, q, t \in \mathbf{Z}^+$, where $t \geq 2$ and k is a prime number such that $k \geq q$, then an $OOA_1(q, t, k, t)$ exists. Thus, if we let $q = k$, we have the following corollary.

Corollary 3.10. *Let $k, t \in \mathbf{Z}^+$, where $t \geq 2$ and k is a prime number. If $(\mathcal{I}, \mathcal{O})$ is a k -partite hypergraph, κ is a base- k function, and $\epsilon(\mathcal{O}, \kappa) = k^t$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has a strongly equitable optimal test suite of size $\epsilon(\mathcal{O}, \kappa)$.*

Unlike Theorem 3.6, we have been unable to use this result to obtain test suites for software systems $(\mathcal{I}, \mathcal{O}, \kappa)$ where $(\mathcal{I}, \mathcal{O})$ is an arbitrary hypergraph.

4 Construction 3 and Conclusion

In Sections 2 and 3, we have presented families of software systems $(\mathcal{I}, \mathcal{O}, \kappa)$ for which optimal test suites can be constructed efficiently. For the first set of families, the criteria for optimality completely relied on the structure of $(\mathcal{I}, \mathcal{O})$ (i.e., whether it is a bipartite graph, a cycle, or a hypertree). For the second set of families, the criteria was a combination of the structure of $(\mathcal{I}, \mathcal{O})$ and the function κ (i.e., whether it is a base- k function for some prime number k). We present a third set of families which, in its simplest version, is able

to produce an optimal test suite because of the function κ only. The proof is remarkably simple.

Theorem 4.1. *If $\kappa(I_s)$ and $\kappa(I_t)$ are relatively prime for every pair of parameters I_s and I_t , then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite with size $\epsilon(\mathcal{O}, \kappa)$ which can be constructed in $O(|\mathcal{I}|\epsilon(\mathcal{O}, \kappa))$ time.*

Proof: Let \mathcal{T} be an $|\mathcal{I}| \times \epsilon(\mathcal{O}, \kappa)$ array whose rows are indexed by the parameters in \mathcal{I} . For the row indexed by $I \in \mathcal{I}$, say row i , let $\mathcal{T}_{ij} = j \bmod \kappa(I)$, $j = 0, \dots, \epsilon(\mathcal{O}, \kappa) - 1$. Thus, \mathcal{T} is clearly equitable and can be constructed in $O(|\mathcal{I}|\epsilon(\mathcal{O}, \kappa))$ time. To prove that it is a valid test suite for $(\mathcal{I}, \mathcal{O}, \kappa)$, consider an arbitrary $O = \{I_{s_1}, I_{s_2}, \dots, I_{s_r}\} \in \mathcal{O}$. Its combination $(t_{s_1}, t_{s_2}, \dots, t_{s_r})$ is covered by \mathcal{T} if there exists a column j such that $t_{s_k} = j \bmod \kappa(I_{s_k})$ for $k = 1, \dots, r$. According to the Chinese Remainder Theorem, such a j exists and lies in the range 0 to $(\prod_{k=1}^r \kappa(I_{s_k})) - 1$ because the integers $\kappa(I_{s_1}), \kappa(I_{s_2}), \dots, \kappa(I_{s_r})$ are pairwise relatively prime. Since \mathcal{T} has $\epsilon(\mathcal{O}, \kappa)$ columns and $\epsilon(\mathcal{O}, \kappa) \geq (\prod_{k=1}^r \kappa(I_{s_k}))$, it follows that \mathcal{T} covers this combination of O . \square

Notice that the proof simply relied on the fact that every pair of parameters I_s and I_t in each $O \in \mathcal{O}$ have $\kappa(I_s)$ and $\kappa(I_t)$ relatively prime. Thus, we can strengthen the theorem as follows.

Corollary 4.2. *If, for every pair of parameters I_s and I_t , $\kappa(I_s)$ and $\kappa(I_t)$ are relatively prime whenever I_s and I_t belong to the same $O \in \mathcal{O}$, then $(\mathcal{I}, \mathcal{O}, \kappa)$ has an equitable test suite with size $\epsilon(\mathcal{O}, \kappa)$ which can be constructed in $O(|\mathcal{I}|\epsilon(\mathcal{O}, \kappa))$ time.*

Aside from finding new optimal instances of TSG, we would also like to emphasize the techniques we used to arrive at the results. Our first construction is graph-theoretic in nature – a departure from the common approaches used in creating orthogonal arrays or covering arrays. Our second construction shows that much stronger results can be obtained by (a) examining the structure of Bush’s technique for creating orthogonal arrays and (b) using ordered orthogonal arrays instead of just orthogonal arrays to create test suites. Finally, our third construction relies on the Chinese Remainder Theorem to create optimal test suites.

Additionally, we have also used our newly found optimal instances to generate test suites for arbitrary software systems. In particular, we proved that when $(\mathcal{I}, \mathcal{O})$ is a graph, $(\mathcal{I}, \mathcal{O}, \kappa)$ for *any* κ has a test suite whose size is at most $\lceil \log_2 n \rceil \times OPT$ that can be constructed in polynomial time. Our bound matches the one guaranteed by the greedy algorithm, which is currently the best known for TSG. Several interesting questions remain: (i) Aside from bipartite graphs, cycles, and hypertrees, what other graph families can $(\mathcal{I}, \mathcal{O})$ belong to so that $(\mathcal{I}, \mathcal{O}, \kappa)$ has an optimal test suite for any κ that can be constructed efficiently? (ii) Similarly, aside from base- k functions, what other classes of functions can κ belong to so that for many hypergraphs $(\mathcal{I}, \mathcal{O})$, optimal test suites for $(\mathcal{I}, \mathcal{O}, \kappa)$ can be constructed optimally? (iii) Finally, can we extend our techniques for constructing test suites of size $O(\log n) \times OPT$ to $(\mathcal{I}, \mathcal{O}, \kappa)$ where $(\mathcal{I}, \mathcal{O})$ is an arbitrary hypergraph?

Acknowledgments

I would like to thank Adrian Dumitrescu, Rongqing Tu, and Jeb Willenbring for discussing parts of this paper with me, and the anonymous referees whose comments have improved the quality of this paper.

References

- [1] K. Bush. Orthogonal arrays of index unity. *Ann. Math. Statist.*, pages 426–434, 1952.
- [2] C. Cheng, A. Dumitrescu, and P. Schroeder. Generating small combinatorial test suites to cover input-output relationships. In *Proc. 3rd Int. Conf. Quality Software*, pages 76–82, 2003.
- [3] D. Cohen, S. Dalal, M. Fredman, and G. Patton. The AETG system: an approach to testing based on combinatorial design. *IEEE Trans. Software Eng.*, 23(7):437–444, 2000.
- [4] M. Cohen, C. Colbourn, J. Collofello, P. Gibbons, and W. Mugridge. Variable strength interaction testing of components. In *Proc. 27th Ann. Intl. Computer Software and Applications Conf.*, pages 413–418, 2003.
- [5] M. Cohen, C. Colbourn, P. Gibbons, and W. Mugridge. Constructing test suites for interaction testing. In *Proc. 25th Intl. Conf. Software Eng.*, pages 38–48, 2003.
- [6] C. Colbourn, M. Cohen, and R. Turban. A deterministic density algorithm for pairwise interaction coverage. In *Proc. IASTED Int. Conf. on Software Eng.*, pages 242–252, 2004.
- [7] A. Dumitrescu. Efficient algorithms for generation of combinatorial covering suites. In *Proc. Int. Symp. Algorithms and Computation*, pages 300–308, 2003.
- [8] H. Faure. Discrepancy of sequences associated with a number system (in dimension s) (in french). *Acta Arithmetica*, 41(4):337–351, 1982.
- [9] M. Grindal, J. Offutt, and S. Andler. Combination testing strategies: A survey. *To appear in Journal of Software Testing, Verification and Reliability*, 2005.
- [10] A. Hartman. Software and hardware testing using combinatorial covering suites. Presented at 2nd Interdisciplinary Applications of Graph Theory, Combinatorics and Algorithms, manuscript, July 2003.
- [11] K. Lawrence. *Combinatorial bounds and constructions in the theory of uniform point distribution in unit cubes, connections with orthogonal arrays and a poset generalization of a related problem in coding theory*. PhD thesis, University of Wisconsin-Madison, 1995.
- [12] Y. Lei and K. C. Tai. In-parameter order: a test generation strategy for pairwise testing. In *Proc. 3rd IEEE High-Assurance Systems Eng. Symp.*, pages 254–261, 1998.

- [13] J. Matousek. *Geometric discrepancy, an illustrated guide*. Springer-Verlag, 1999.
- [14] K. Meagher, L. Moura, and L. Zekaoui. Mixed covering arrays on graphs. *Journal of Combinatorial Designs*. To appear.
- [15] K. Meagher and B. Stevens. Covering arrays on graphs. Submitted to *J. Combin. Theory Ser. B*, 2002.
- [16] L. Moura, B. Stevens, J. Stardom, and A. Williams. Covering arrays with mixed alphabet sizes. *J. Combin. Des.*, 11(6), 2003.
- [17] H. Niederreiter. Constructions of (t, m, s) -nets and (t, s) sequences. *Finite Fields and Their Applications*, 11:578–600, 2005.
- [18] W. Schmid. *(T,M,S)-nets: digital constructions and combinatorial aspects*. PhD thesis, Universität Salzburg, 1995.
- [19] P. Schroeder and B. Korel. Black-box test reduction using input-output analysis. In *Proc. Int. Symp. Software Testing and Analysis*, pages 21–24, 2000.
- [20] G. Seroussi and N. H. Bshouty. Vector sets for exhaustive testing of digital circuits. *IEEE Trans. Information Theory*, 34(3):513–522, 1988.
- [21] T. Yu-Wen and W. Aldiwan. Automating test case generation for the new generation of mission software system. In *Proc. IEEE Aerospace Conf.*, pages 431–437, 2000.

Appendix

We transform the proof of Theorem 2.3 to algorithm GENERATE-TSG. In the algorithm, $(V_1 \cup V_2, E)$ is a connected bipartite graph, and the test suite \mathcal{T} is stored in an $n \times \epsilon(\mathcal{O}, \kappa)$ array whose rows are indexed by I_1, I_2, \dots, I_n . Each time GENERATE is called, at least one row in \mathcal{T} is filled.

GENERATE-TSG(V_1, V_2, E, κ)

Create \mathcal{T} , an $n \times \epsilon(\mathcal{O}, \kappa)$ array whose rows are indexed by the parameters in $V_1 \cup V_2$.

Initialize all entries of \mathcal{T} to 0.

$\mathcal{T} \leftarrow$ GENERATE($V_1, V_2, E, \kappa, \mathcal{T}$)

GENERATE($V_1, V_2, E, \kappa, \mathcal{T}$)

$I_z \leftarrow$ an input parameter in $V_1 \cup V_2$ with largest number of data values

$I_y \leftarrow$ an input parameter in $N(I_z)$ with largest number of data values

if $|V_1| = |V_2| = 1$

 for $v = 0$ to $\kappa(I_y) - 1$

 for $j = v \times \kappa(I_z)$ to $v \times \kappa(I_z) + \kappa(I_z) - 1$

$\mathcal{T}[I_y][j] \leftarrow v$

$v \leftarrow 0$

```

for  $j = 0$  to  $\kappa(I_y)\kappa(I_z) - 1$ 
   $\mathcal{T}[I_z][j] \leftarrow v$ 
   $v \leftarrow (v + 1) \bmod \kappa(I_z)$ 
else
  if  $|N(I_z)| = 1$  /*  $I_z$  has only one neighbor */
    Delete  $I_z$  from  $V_1 \cup V_2$  and  $(I_y, I_z)$  from  $E$ 
     $\mathcal{T} \leftarrow \text{GENERATE}(V_1, V_2, E, \kappa, \mathcal{T})$ 
     $M \leftarrow \max\{\epsilon(E, \kappa), \kappa(I_y) \times \kappa(I_z)\}$ 
    if  $M > \epsilon(E, \kappa)$ 
      for  $i = 1$  to  $M - \epsilon(E, \kappa)$ 
        Using  $\text{ADD}(\mathcal{T})$ , add a test case for the system  $(V_1 \cup V_2, E, \kappa)$  to  $\mathcal{T}$  at
        column  $\epsilon(E, \kappa) - 1 + i$ 
      Sort the numbers  $\mathcal{T}[I_y][j]$ ,  $j = 0, \dots, M - 1$  and store the column numbers of the
      sorted numbers in array  $A$  so that  $\mathcal{T}[I_y][A[j]] \leq \mathcal{T}[I_y][A[j + 1]]$  for  $j = 0, \dots, M - 2$ 
       $v \leftarrow 0$ 
      for  $j = 0$  to  $M - 1$ 
         $\mathcal{T}[I_z][A[j]] \leftarrow v$ 
         $v \leftarrow (v + 1) \bmod \kappa(I_z)$ 
    else /*  $I_z$  has more than one neighbor */
       $X \leftarrow N(I_z) - \{I_y\}$ 
      Contract  $N(I_z)$  to  $I_y$  and update  $(V_1 \cup V_2, E, \kappa)$  accordingly
       $\mathcal{T} \leftarrow \text{GENERATE}(V_1, V_2, E, \kappa, \mathcal{T})$ 
      for each  $I_s \in X$ 
        for  $j = 0$  to  $\epsilon(E, \kappa) - 1$ 
           $\mathcal{T}[I_s][j] \leftarrow \mathcal{T}[I_y][j]$ 
        Apply  $\text{MODIFY}(I_s, \kappa(I_y), \kappa(I_s))$  to  $\mathcal{T}$  but only up to column  $\epsilon(E, \kappa) - 1$ .
return( $\mathcal{T}$ )

```

Theorem 4.3. *Algorithm TSG-GENERATE creates an equitable test suite \mathcal{T} for the software system $(V_1 \cup V_2, E, \kappa)$ in $O(nm + n\epsilon(\mathcal{O}, \kappa))$ time where $n = |V_1 \cup V_2|$ and $m = |E|$.*

Proof: The correctness of TSG-GENERATE follows directly from Theorem 2.3. The two modifications we made occur when I_z has only one neighbor: (1) Instead of explicitly sorting the columns $\mathcal{T}[*][0], \mathcal{T}[*][1], \dots, \mathcal{T}[*][M - 1]$ based on their I_y -values, we simply sort the numbers $\mathcal{T}[I_y][0], \mathcal{T}[I_y][1], \dots, \mathcal{T}[I_y][M - 1]$ and store the column numbers of the sorted numbers in array A so that the columns do not have to be moved. (2) Instead of following the steps in Lemma 2.1 to create the row for I_z , we add a row for I_z so that $\mathcal{T}[I_z][A[0]], \mathcal{T}[I_z][A[1]], \dots, \mathcal{T}[I_z][A[M - 1]]$ consist of sequences of the permutation $0, 1, \dots, \kappa(I_z) - 1$. Since each I_y -value occurs at least $\kappa(I_z)$ times because $M \geq \kappa(I_y)\kappa(I_z)$, every combination of (I_z, I_y) is covered by \mathcal{T} .

Each time GENERATE is called, there are three types of work performed: (a) rows and/or columns of \mathcal{T} are filled, (b) I_z and I_y are found and the graph $(V_1 \cup V_2, E)$ modified either by deleting vertices or contracting a set of vertices, and (c) pre-processing and/or post-processing work is done on \mathcal{T} before or after the entries of \mathcal{T} are filled.

Let $n = |V_1 \cup V_2|$ and $m = |E|$. The total time spent on type (a) work is proportional to the size of the array \mathcal{T} , $n\epsilon(\mathcal{O}, \kappa)$. Finding I_z and I_y takes $O(n)$ time; modifying $(V_1 \cup V_2, E)$

requires $O(n + m)$ time. The pre-processing work of sorting the I_y -values of the columns of T takes $O(\epsilon(\mathcal{O}, \kappa))$ time by using bucket sort. The post-processing work of modifying I_s 's value from $\kappa(I_y)$ to $\kappa(I_s)$ takes $O(\epsilon(\mathcal{O}, \kappa))$ time. Whenever GENERATE is called, at least one parameter's row (e.g., I_z if $|N(I_z)| = 1$, every $I_s \in X$ if $|N(I_z)| > 1$) is filled so there are at most n calls to GENERATE. Hence, the total time spent on types (b) and (c) work is $O(n^2 + nm + n\epsilon(\mathcal{O}, \kappa))$ time. The theorem follows. \square

Next, we present the proof for Lemma 3.5.

Lemma 3.5. *Let $k = p^z$ where p is a prime number and z is a positive integer. Let A be an $OA_1(k, k, 2)$ orthogonal array obtained by Bush's construction. Let $A[r]$ denote the array obtained by replacing each α_i by $\alpha_{i \bmod r}$ in the entries of A . The elements of $GF(k)$ can be chosen and the columns of A can be arranged so that for each non-negative integer $z' \leq z$, every row of $A[p^{z'}]$ is a concatenation of permutations of $\{\alpha_0, \alpha_1, \dots, \alpha_{p^{z'}-1}\}$.*

Proof: Recall that every element of $GF(k)$ can be written as a polynomial $a_0 + a_1x + a_2x^2 + \dots + a_{z-1}x^{z-1}$ where the coefficients are elements of $\{0, 1, \dots, p-1\}$. Denote such an element by α_i if $i = a_0 + a_1p + a_2p^2 + \dots + a_{z-1}p^{z-1}$, and define a total order on the elements of $GF(k)$ by letting $\alpha_i < \alpha_{i'}$ whenever $i < i'$. Hence, $\alpha_0 = 0$, $\alpha_1 = 1$, etc. Next, arrange the columns of A so that its first k columns are indexed by the polynomials $\alpha_0x + \alpha_0, \alpha_0x + \alpha_1, \dots, \alpha_0x + \alpha_{k-1}$, the next k ones by $\alpha_1x + \alpha_0, \alpha_1x + \alpha_1, \dots, \alpha_1x + \alpha_{k-1}$, the next k ones by $\alpha_2x + \alpha_0, \alpha_2x + \alpha_1, \dots, \alpha_2x + \alpha_{k-1}$, etc. That is, the vector of coefficients of the polynomials are lexicographically increasing.

If we partition A into subarrays of $p^{z'}$ columns each, where $z' \leq z$ is an integer, it is straightforward to verify that because $k = p^z$ the columns within each subarray are indexed by polynomials of the form $\phi + \alpha_0, \phi + \alpha_1, \dots, \phi + \alpha_{p^{z'}-1}$ where ϕ is some polynomial over $GF(k)$. When these polynomials are evaluated at some element of $GF(k)$, the results form a sequence of the form $\alpha_i + \alpha_0, \alpha_i + \alpha_1, \dots, \alpha_i + \alpha_{p^{z'}-1}$ for some $\alpha_i \in GF(k)$. Because of the way we have denoted the elements of $GF(k)$, this same sequence must then be a permutation of the elements in $\{\alpha_{qp^{z'}}, \alpha_{qp^{z'}+1}, \dots, \alpha_{qp^{z'}+p^{z'}-1}\}$ where $q = \lfloor i/p^{z'} \rfloor$. This implies that every row of $A[p^{z'}]$ is a concatenation of permutations of $\{\alpha_0, \alpha_1, \dots, \alpha_{p^{z'}-1}\}$. See an example in Figure 2. \square