Combinatorial Designs to Explore Large Experimental Search Spaces *

Gary L. Mullen †Dennis Shasha ‡Joseph L. Yucas §

April 4, 2005

Abstract

Genomic and proteomic studies take advantage of omic level techniques such as microarrays to achieve species-wide scale. Obtaining an operational model (e.g., a virtual animal) however requires a search in a space consisting of many factors and many values of each factor, all of which may interact. This suggests the need to perform a disciplined search in that space. We explain the use of and some new results in covering arrays, a technique from the theory of combinatorial designs, and explain how to use covering arrays at various stages of experiment and analysis.

1 Introduction:

Your favorite organism can be grown under several conditions by varying light, food, water, and various nutrients. Several natural questions suggest themselves:

*Work supported in part by U.S. NSF grants IIS-9988345, N2010-0115586 and MCB-0209754.

†Department of Mathematics, The Pennsylvania State University, University Park, PA 16802, USA, Email: mullen@math.psu.edu

‡Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, Email: shasha@cs.nyu.edu

§Department of Mathematics, Southern Illinois University, Carbondale, IL 62901, Email: jyucas@math.siu.edu

- Which conditions give rise to the best growth, either of the organism as a whole or of some protein of interest?
- Which factors are the most critical?
- Which genes or proteins react most strongly to some factor or collection of factors?

Given unlimited resources and time, you would want to test all possible conditions by testing all values of every factor. Unfortunately, this very soon becomes a daunting task. For example, 10 inputs each having four possible values generates somewhat more than one million experiments. This may be more than most labs can or want to do. Further many of these conditions may yield very similar results to one another because they may differ only in unimportant factors.

Covering arrays, a technique from the theory of combinatorial designs, constitute a disciplined sampling method that give certain coverage guarantees on the search space of conditions while generating few experiments. Before defining the concept formally, let us give an example.

Suppose that we have six factors A, B, C, D, E, F that we can manipulate, each having three values $0, 1, 2$. Covering the entire search space requires $3^6 = 729$ tests. In that case, each of the possible 6-tuples based upon these three values would appear exactly once in the array.

What would a disciplined sampling approach require? First every possible value of every input factor should be present in some experiment. If this were the only condition, then it could be satisfied with the following three experiments:

Exp	A	B	C	D	E	F
1	0	0	0	0	0	0
2	1	1	1	1	1	1
3	2	2	2	2	2	2

.

Intuitively, this is unsatisfactory, because it tests no interactions at all. So, let us raise the bar a little. We want to cover every value of every factor as above, but also for every pair of factors, we want to test each possible pair of values. In this way, if two values of two factors entirely dominate the situation, an experiment will discover that condition. Because every pair of factors has 9 possible pairs of values, this gives a lower bound of 9 experiments altogether, but it might seem to be difficult to come close to this few.

Remarkably, one can. A mere 13 experiments is enough:

Exp	A	B	C	D	E	F
1	0	0	0	0	0	0
2	0	1	1	1	1	2
3	0	2	2	2	2	0
4	0	0	1	2	0	1
5	1	0	0	1	1	1
6	1	1	2	2	0	1
7	1	2	1	0	2	1
8	1	1	2	0	1	0
9	2	0	2	1	2	2
10	2	1	0	0	2	1
11	2	2	1	1	0	0
12	2	2	0	2	1	2
13	1	0	0	0	0	2

In what sense does this cover every pair of values for each pair of factors? Consider B and E for example. Let us project onto their two columns:

B	E
0	0
1	1
2	2
0	0
0	1
1	0
2	2
1	1
0	2
1	2
2	0
2	1
0	0

The first three rows cover the pairs $(0,0), (1,1)$ and $(2,2)$. The fourth row repeats $(0,0)$ for that pair, but then we get, in succession, $(0,1), (1,0)$. After more repeats, we get the rest: $(0,2), (1,2), (2,0)$, and $(2,1)$. What is clever about this construction is that while taking care of factors B and E we are also taking care of all other pairs of factors.

Let's review what this 13 experiment design accomplishes. Every value of every factor is tested. Every pair of values of every pair of factors is also tested. Of course, most three and four way interactions are not tested. There is no free lunch. But if you have limited resources, such coverage reveals a lot in few experiments [9]. If you have more resources, then you can do more experiments and cover more interactions. We sketch some of the technicalities below.

Before continuing with the mathematical development, we discuss the variety of ways in which this might apply to biology. First, the working biologist may want to know which factors are likely to be most important in their regulation of some target such as organism growth. Analyzing the results of an experimental set like the one above may uncover strong correlations with one factor or another. Those

correlations are not definitive (there are many confounding factors), but they suggest factors which may have a strong effect. The next step is to test the importance of those factors.

Second, suppose you have discovered some factor that seems important. Or alternatively, suppose that you want to focus on that factor in an undirected mode, e.g., to discover which genes or proteins were sensitive to that factor. In either case you could take each value of that factor (or factors) and append a covering array of the other factors. In this example, suppose that we wanted to check whether factor A has a consistent effect in every context. The following 36 element array tests each value of A against a widely varying backgroound of the other inputs:

Exp	A	B	C	D	E	F
1	0	0	0	0	0	0
2	0	0	1	1	1	1
3	0	0	2	2	2	2
4	0	0	0	1	2	0
5	0	1	0	0	1	1
6	0	1	1	2	2	0
7	0	1	2	1	0	2
8	0	1	1	2	0	1
9	0	2	0	2	1	2
10	0	2	1	0	0	2
11	0	2	2	1	1	0
12	0	2	2	0	2	1
13	1	0	0	0	0	0
14	1	0	1	1	1	1
15	1	0	2	2	2	2
16	1	0	0	1	2	0
17	1	1	0	0	1	1
18	1	1	1	2	2	0
19	1	1	2	1	0	2
20	1	1	1	2	0	1
21	1	2	0	2	1	2
22	1	2	1	0	0	2
23	1	2	2	1	1	0
24	1	2	2	0	2	1
25	2	0	0	0	0	0
26	2	0	1	1	1	1
27	2	0	2	2	2	2
28	2	0	0	1	2	0
29	2	1	0	0	1	1
30	2	1	1	2	2	0
31	2	1	2	1	0	2
32	2	1	1	2	0	1
33	2	2	0	2	1	2
34	2	2	1	0	0	2
35	2	2	2	1	1	0
36	2	2	2	0	2	1

You will notice that the three groups 1-12, 13-24, and 25-36 are the same except in their A value. Thus for example 3, 15 (3+12), and 27 (3+24) have the same values of B, C, D, E, F but differ in their A values. So for each A value, all pairwise interactions among B, C, D, E, F are tested (by 12 tests for these five factors rather than the 13 tests required for six factors). Therefore, all three-way interactions involving A are tested overall. If for each experiment triple i, $i + 12$, $i + 24$, the value of some target increases, then it is likely that A is inductive for that target. Admittedly, this notion of likelihood is difficult to quantify, because we don't know the underlying distribution of data, but the construction gives such a large variety of values of B, C, D, E, and F that this statement is still an excellent hypothesis.

But what if A has no such consistent pattern. This brings us to a third use of combinatorial design. Suppose for example that in this last array, experiment 28 gave a strong inductive effect relative to 16 (28-12) and 4 (28-24), whereas 25 did not have a strong inductive effect relative to 13 and 1. Presumably the interaction with B, C, D, E, F is responsible. If we look at those values in the two cases, we see that 25 and 28 (and therefore pairs $(1, 3)$ and $(13, 16)$) share the same value for B, C, and F. On the other hand, they differ in D and E. So one could establish a context that mixes the two, e.g. $D = 0$ and $E = 2$, and pose the three experiments:

Exp	A	B	C	D	E	F
$1'$	0	0	0	0	2	0
$2'$	1	0	0	0	2	0
$3'$	2	0	0	0	2	0

If the effect is strongly inductive, then factor E is critically important in the influence of A. This ability to identify "intermediate" experiments and to zero in on critical factors can make experimental practice far more efficient. We have used this successfully in the study of nitrogen pathways in Arabidopsis [12, 9].

2 Mathematical Definitions and Results

The purpose of this section is to help you understand basic definitions and the number of experiments you would need depending on the number of factors you have and the number of values per factor. In most of the development, we assume the same number of values per factor, but if you have fewer than the maximum for some factor, the software [14] may be able to take advantage of that to reduce the number of needed experiments.

2.1 Covering Arrays

Let us say there are c factors, v values and we want to cover all possible t-way interactions. In the 13 experiment example above, there were $c = 6$ factors, each having $v = 3$ values, and we were interested in $t = 2$ way interactions. Here is the general definition.

A *t-covering array* with alphabet size v, length c, and size r consists of r vectors of length c with entries from $\{0, 1, \ldots, v-1\}$ with the property that in any t columns of the array each of the v^t ordered t-tuples occurs at least once [13]. In this terminology, our earlier examples are 2-covering arrays.

The main problem in designing these covering arrays is to minimize r (the number of rows or experiments) for given values of v, c, t, or equivalently, to maximize c (the number of factors) for given values of v, r, t.

In the case when $v = t = 2$ (two values per factor and two-way interactions), the problem has been completely solved [13]. In particular, for any fixed value of r, the maximal number c of factors is determined by

$$c = \binom{r-1}{\lceil \frac{r}{2} \rceil}.$$

At first glance, this may not seem helpful. Working biologists will often be given

a number of factors and want to find out how many experiments are needed, rather than being given a number of experiments and finding out how many factors can be tested. The formula lets us figure out the number of experiments from the number of factors however. The following table shows how many experiments are needed for each number of conditions for this (two value, two-way interaction) case:[1]

c	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
r	4	4	5	6	6	6	6	6	6	7	7	7	7	7	8	8	8	8	8

For large c, using logs base 2, the number of rows is $r = logc + \frac{1}{2}loglogc \ldots$.

For $t = 2, v = 3$ (two-way interactions but three values per factor), a small table of the best known values taken from [13] follows:

c	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
r	9	9	9	11	12	15	15	15	15	15	15	18	18	18	18	21	21

For $t = 2, v > 2$, it is known that for large c the minimal number of rows satisfies $r = \frac{v}{2}log(c(1 + o(1)))$. Thus it rises as the log of the number of conditions rather than exponentially (which would be the case for a complete search of the entire space).

In [11] the authors study covering arrays and their constructions in more detail. In particular, using sets of mutually orthogonal latin squares, they provide constructions of covering arrays which improve a number of the currently best known parameters for covering arrays; including three cases ($c = 13, r = 15; c = 14, r = 15;$ and $c = 15, r = 15$) in the above table for $v = 3$. See [3], [4], or [8] for a discussion of latin squares and sets of mutually orthogonal latin squares.

[1]Such a covering array can be constructed by assuming the first row consists of all zeros, and the remaining $r - 1$ rows are taken to be the characteristic vectors of all subsets of weight $\lceil \frac{r}{2} \rceil$ of a set containing $r - 1$ elements.

Using our software [14], for example, computing 2-way interactions for 10 factors each with 3 values requires 15 experiments (as opposed to $3^{10} = 59,049$ to explore the complete space), and computing 2-way interactions for 10 factors each with 4 values requires 52 experiments (as compared with $4^{10} = 1,065,024$ to explore the complete space).

What follows is an optimal covering array for two-way interactions ($t = 2$) on six factors ($c = 6$), with three values ($v = 3$) per factor. Recall from our earlier example that we had 13 experiments, now we have $r = 12$ experiments, and as indicated from the $v = 3$ table, this value of r cannot be decreased.

A	B	C	D	E	F
0	0	1	2	2	1
0	1	0	1	2	2
0	2	1	0	1	2
0	2	2	1	0	1
0	1	2	2	1	0
1	1	1	0	0	0
1	0	0	1	1	0
1	2	0	2	0	1
1	0	2	0	2	2
2	0	2	2	0	2
2	1	0	0	1	1
2	2	1	1	2	0

2.2 Orthogonal Arrays

Whereas we think that covering arrays provide the most parsimonious approach to testing interactions, reasonable scientists might desire a property known as balance. For this we need *orthogonal arrays*.

An orthogonal array [7] has, like a covering array, r rows (experiments) and c (factors) columns. Again, assuming that each factor has v values, the array has "strength" t (number of interacting factors) and index λ if every $r \times t$ subarray

(projection of every t columns) of A contains each t-tuple in exactly λ rows. (Thus an orthogonal array is a covering array, but not every covering array is an orthogonal array since the pairs may not all occur the same number of times in the covering array.)

For example, for an orthogonal array of strength $t = 2$, each of the v elements occur the same number of times in each column, but each of the v^2 possible ordered pairs also occurs the same number of times in any two columns. We also note that an orthogonal array is optimal (with a minimal number r of rows) and can be constructed for all prime power values of v provided $c \leq v + 1$, see for example [3] or [8].

The following is an example of an orthogonal array based upon $v = 3$ symbols with $r = 9$ rows, $c = 4$ columns, and of strength $t = 2$ with index $\lambda = 1$.

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 2 & 0 \\ 1 & 2 & 0 & 1 \\ 2 & 0 & 2 & 1 \\ 2 & 1 & 0 & 2 \\ 2 & 2 & 1 & 0 \end{array}$$

This array arises from the following pair of orthogonal latin squares of order 3, which form the last two columns of the array:

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

Chapter 12 of [7] provides an excellent summary of orthogonal arrays and how to construct them when they are possible to construct. When they aren't, *main-effects*

plans may be used. A main-effects plan is intermediate between a covering array which guarantees nothing about balance and an orthogonal array. In a main-effects plan, the number of occurrences of each ordered is proportional to the number of times that each element appears in each column. For a discussion of main-effect plans, see Section 11.7 of [7]. The website www.research.att.com/ njas/ contains recent updates on orthogonal arrays and their constructions.

3 Summary:

Combinatorial designs in general and covering arrays in particular can vastly reduce the number of experiments needed to explore a search space. While they don't cover every possible combination of the input factors, they sample the search space in a well-separated manner and guarantee that every combination of values in every t factors are tested. If a researcher prefers certain balance properties, then he or she can use orthogonal arrays or main-effects orthogonal arrays.

Combinatorial designs can be used in an iterative and adaptive fashion.

1. One can start by using it to find those factors that might be important.

2. Given a possibly important factor, one can use combinatorial design to test the consistency of its importance.

3. Given one set of conditions where a factor is important and another set where the factor isn't, one can create intermediates that will isolate what it is about a context that determines the importance of the factor.

Of course, t-way covering arrays don't test $(t+1)$-way interactions. Thus, to extend one's interactivity coverage, one must do more work. But given a certain desired coverage, covering arrays give an inexpensive way to uncover them.

Acknowledgments: The second author acknowledges his excellent collaboration with Gloria Coruzzi and her excellent research group whose needs drove the development of adaptive combinatorial designs [9]. The second author first saw combinatorial designs in use at Telcordia where it was used for software testing and which provides excellent software [1] Terry Dwyer started this collaboration.

References

[1] Cohen D.M., Dalal S.R., Fredman M. L., Patton G.C., The AETG System: An Approach to Testing Based on Combinatorial Design. IEEE Transactions On Software Engineering, 1997. 23, 437-444.

[2] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein, Covering Codes, North-Holland Mathematical Library, Elsevier, Amsterdam, 1997.

[3] J. Dénes and A.D. Keedwell, Latin Squares and Their Applications, Academic Press, New York, 1974.

[4] J. Dénes and A.D. Keedwell, Latin Squares, Annals of Disc. Math., Vol. 46(1991).

[5] R.H. Hardin and N.J.A. Sloane, Operating Manual for Gosset: A General-Purpose Program for Constructing Experimental Designs, Sec. ed., 1992, Statist. Dept. Report 106, AT&T Labs, Murray Hill, NJ.

[6] R.H. Hardin and N.J.A. Sloane, *A new approach to the construction of optimal designs*, J. Statist. Plann. Infer. 37(1993), 339-369.

[7] A.S. Hedayat, N.J.A. Sloane, and J. Stufken, Orthogonal Arrays: Theory and Applications, Springer Verlag, New York, 1999.

[8] C.F. Laywine and G.L. Mullen, Discrete Mathematics using Latin Squares, Wiley, New York, 1998.

[9] L.V. Lejay, D.E. Shasha, A.Y. Kouranov, P.M. Palenchar, A.A. Cruikshank, M.F. Chou, and G.M. Coruzzi, *Adaptive Combinatorial Design: A Tool for Systems Biology*, PNAS, 2003. under review.

[10] G.L. Mullen, *Polynomial representation of complete sets of mutually orthogonal frequency squares of prime power order*, Disc. Math. 69(1988), 79-84.

[11] G.L. Mullen, D. Shasha, and J.L. Yucas, *Covering arrays*, preprint.

[12] 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(2001), 1590-1594.

[13] N.J.A. Sloane, *Covering arrays and intersectiong codes*, J. Combin. Designs 1(1993), 51-63.

[14] Dennis Shasha's covering array software: http://cs.nyu.edu/cs/faculty/shasha/papers