A New Algorithm for Constructing Orthogonal and Nearly-Orthogonal Arrays

R. A. Lekivetz, D. Bingham, and R. R. Sitter
Simon Fraser University, Burnaby, BC, V5A 1S6, Canada
M.S. Hamada, L.M. Moore, and J.R. Wendelberger
Los Alamos National Laboratory, Los Alamos, NM, 87545

Abstract

Orthogonal arrays are frequently used in industrial experiments for quality and productivity improvement. Due to run-size constraints and level combinations, an orthogonal array may not exist, in which case a nearly-orthogonal array can be used. Orthogonal and nearly-orthogonal arrays can be difficult to find. This poster will introduce a new algorithm for the construction of orthogonal arrays and nearly-orthogonal arrays with desirable statistical properties, and compare the new algorithm to a pre-existing algorithm.

Introduction

Experimenters are often interested in studying a number of factors in a small number of runs. One way to do so is through the use of orthogonal and nearly-orthogonal arrays.

For a general factorial design, we consider the standard normal regression model for a design $\mathbf{d}$,

$$Y = X_0\alpha_0 + X_1\alpha_1 + \cdots + X_m\alpha_m + \epsilon.$$

- Y is the vector of observations.
- α_j the vector of j-factor interactions.
- X_j the matrix of coefficients for α_j (column i corresponds to the coefficient for the ith effect).
- ϵ iid $N(0, \sigma^2)$.

How do we measure 'near' orthogonality?

A number of different approaches have been taken.

Xu and Wu [2] defined $A_j(\mathbf{d})$ as a measure of the aliasing between the j-factor interactions and the general mean. For $X_j = [x_{ik}^{(j)}]$, let

$$A_j(\mathbf{d}) = \frac{1}{N^2}\sum_k \left|\sum_{i=1}^{N} x_{ik}^{(j)}\right|^2.$$

- A_j measures aliasing between j-factor interactions and mean.
- Generalized minimum aberration sequentially minimizes $(A_1, A_2, A_3, \ldots)$.
- $A_2 = 0$ if the design is an orthogonal array.
- An A_2-optimal design will minimize A_2 - our measure of near-orthogonality.

For designs with balanced columns, two equivalent measures of A_2 for a design $\mathbf{d}$ are $ave(\chi^2(\mathbf{d}))$ and $J_2(\mathbf{d})$.

Define

$$\chi^2_{kl}(\mathbf{d}) = \sum_{a=0}^{s_k-1}\sum_{b=0}^{s_l-1}\frac{[n_{kl}(a,b) - N/(s_k s_l)]^2}{N/(s_k s_l)},$$

where column l has s_l levels, and $n_{kl}(a,b)$ is the number of times the level combination (a,b) occurs in columns k and l, Ye and Sudjianto [3] use

$$ave(\chi^2(\mathbf{d})) = \sum_{1\le k<l\le m}\chi^2_{kl}(\mathbf{d})/[m(m-1)/2]. \qquad (1)$$

Define

$$\delta_{i,j}(\mathbf{d}) = \sum_{k=1}^{n} w_k\delta(x_{ik}, x_{jk}), 1 \le i, j \le N,$$

where $\delta(a,b) = 1$ if $a = b$, 0 otherwise, w_k is the weight of the column, and $\delta_{i,j}(\mathbf{d})$ is a measure of the similarity between these rows. Then

$$J_2(\mathbf{d}) = \sum_{1\le i<j\le N}[\delta_{i,j}(\mathbf{d})]^2 \qquad (2)$$

$ave(\chi^2(\mathbf{d}))$ is a summation over all columns while $J_2(\mathbf{d})$ is a summation over all rows.

Balanced designs - minimizing (1) or (2) minimizes A_2.

Algorithms

Xu's algorithm [1]

- Sequentially add columns to a design.
- Adds a random balanced column.
- Look at all possible switches in new column, make best one.
- Try adding new column R times - choose best.
- Uses J_2 criterion.

- Call R the number of restarts.

The New Algorithm

- Sequentially adds columns as well.
- New column created one element(row) at a time.
- Uses the χ^2 criterion.
- Can ensure balance is maintained.

Define

$$d^{(h)}_{lb^*} = [x_1^{(h)}|x_2^{(h)}|\cdots|x^{(h)}_{lb^*}],$$

the first h rows, where $x^{(h)}_{lb^*} = (x_{1l}, \cdots, x_{(h-1)l}, b^*)'$.
i.e. b^* is in row h of column l.

Denote $\chi_l^{2(hb^*)}$ as the criterion evaluated with $d^{(h)}_{lb^*}$ for $b^* = 1, \ldots, s_l$:

$$\chi^{2(hb^*)}_{kl} = \chi^{2(h-1)}_{kl} + 2[n^{(h-1)}_{kl}(x_{hk}, b^*) - N/(s_k s_l)] + 1.$$

Considering all columns in the design,

$$\chi^{2(h)}_l = \sum_{k=1}^{l-1}\chi^{2(hb^*)}_{kl}.$$

The algorithm proceeds as follows:

1. Specify an initial design $\mathbf{d}$ with columns $(0, \cdots, 0, 1, \cdots, 1, \cdots, s_1 - 1, \cdots, s_1 - 1)$ and $(0, \cdots, s_2 - 1, 0, \cdots, s_2 - 1, \cdots, 0, \cdots, s_2 - 1)$.
2. For $l = 3, \ldots, n$, do the following:
   i. Randomize the rows of $\mathbf{d}$. Let $h = 1$.
   ii. Let $d^{(h)}_{lb^*}$ be the first h rows, where $x^{(h)}_{lb^*} = (x_{1l}, \cdots, x_{(h-1)l}, b^*)'$.
   iii. For $b^* = 0, \cdots, s_l - 1$, calculate $\chi_l^{2(hb^*)}$. Use the best b^* such that $n_{kl}(a, b^*) \le N/(s_k s_l)$ for $k = 1, \cdots, l-1$. If no such choice exists, take the best b^* with $n_{kl}(a, b^*) > N/(s_k s_l)$. In the case of equally good choices, take the largest or randomly choose between them.
   iv. Repeat Steps ii.-iii. for $h = 1, \cdots, N$.
   v. If $\chi^2(\mathbf{d}) = 0$ go to vii.
   vi. Repeat *i. - v.* R times and choose the best $\mathbf{c}$ which minimizes $\chi^2(\mathbf{d}_+)$.
   vii. Add column $\mathbf{c}$ as the lth column of $\mathbf{d}$.
3. Return the final $N \times n$ design $\mathbf{d}$.

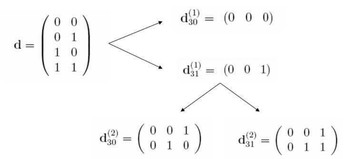

Figure 1: Illustration of new algorithm.

- Want to make best choice for each row.
- Not using all possible switches - saves time.
- Can keep track if expected value is exceeded.
- χ^2 is influenced by number of columns - J_2 not.
- Xu's algorithm also adapted for χ^2 - changes speed?

Results and Conclusions

Orthogonal Arrays: A simulation study was performed on mixed-level orthogonal arrays with small runs using various number of restarts. Table 1 compares the algorithms in terms of best expected time to find an OA (# time OA found / total time spent).

Table 1. Best expected time (in secs) to OA for each algorithm.

OA	New	Xu-χ^2	Xu-J_2	Best	Algorithm
$OA(20, 2^{19})$	0.01773	0.01335	0.01149	0.01149	Xu-J_2
$OA(16, 2^{15})$	0.00217	0.00162	0.00115	0.00115	Xu-J_2
$OA(16, 8^1 2^8)$	0.00037	0.00079	0.00086	0.00037	New
$OA(16, 4^5)$	0.00210	0.02012	0.03428	0.00210	New
$OA(18, 6^1 3^6)$	0.01359	0.02925	0.04579	0.01359	New
$OA(20, 5^1 2^8)$	0.02391	0.02064	0.02689	0.02064	Xu-χ^2
$OA(24, 2^{23})$	0.19994	0.10361	0.09150	0.09150	Xu-J_2
$OA(24, 4^1 2^{20})$	0.08880	0.05605	0.05414	0.05414	Xu-J_2
$OA(24, 3^1 2^{16})$	1.98565	0.72308	0.73488	0.72308	Xu-χ^2
$OA(24, 12^1 2^{12})$	0.00325	0.01045	0.01193	0.00325	New
$OA(24, 4^1 3^1 2^{13})$	0.88230	0.42285	0.48003	0.42285	Xu-χ^2
$OA(25, 5^6)$	0.01844	0.12614	0.33263	0.01844	New
$OA(27, 9^1 3^9)$	0.02161	0.02726	0.05068	0.02161	New
$OA(27, 3^{13})$	129.03000	23.34500	41.72750	23.34500	Xu-χ^2
$OA(28, 2^{27})$	56.31000	5.93750	6.52250	5.93750	Xu-χ^2
$OA(32, 16^1 2^{16})$	0.02403	0.10725	0.12935	0.02403	New
$OA(32, 8^1 4^2 2^{18})$	0.49383	0.15462	0.23916	0.15462	Xu-χ^2
$OA(40, 20^1 2^{20})$	0.26325	1.15383	1.62917	0.26325	New

Nearly-Orthogonal Arrays: To consider NOAs, looked at A_2 for best designs found with the new algorithm with 300, 500, and 1000 restarts and compare this to those found by Xu[1].

- Designs found comparable to Xu's in terms of A_2.
- Order to add columns has impact:
  - $NOA(12, 2^7 3^2)$ finds $A_2 = 0.861$.
  - $NOA(12, 3^2 2^7)$ finds $A_2 = 0.792$.
- Usually best to start with higher-level columns.
- Most designs found within seconds.

Recommendations: The new algorithm tends to work better when the number of factors is small relative to the run size - in these situations, Xu's algorithm with the χ^2 criterion shows an improvement as well. Although Xu's algorithm performs good with 50-100 restarts compared to 500-1000 for the new, a restart with the new algorithm is much faster.

Conclusion: The new algorithm performs well overall in constructing both orthogonal and nearly-orthogonal arrays. There is no clear winner between the new algorithm and Xu's algorithm - sometimes we see an improvement, sometimes not. A thorough discussion is avaiable [4].

References

1. Xu, H. (2002). "An Algorithm for Constructing Orthogonal and Nearly-orthogonal Arrays With mixed levels and small runs ", *Technometrics*, **44**, 356-368.
2. Xu, H., and Wu, C.F.J. (2001). "Generalized Minimum Aberration for Asymmetrical Fractional Factorial Designs", *The Annals of Statistics*, **29**, 549-560.
3. Ye, K. and Sudjianto, A. (2003). "The Use of Cramer V^2 Optimality for Experiments with Qualitative Levels", under revision, submitted to IIE Transactions.
4. Lekivetz, R. (2006). "A New Algorithm for Obtaining Mixed-Level Orthogonal and Nearly-Orthogonal Arrays", M.Sc. Thesis, Dept. of Statistics and Actuarial Science, Simon Fraser University.