

# Optimal Designs with Non-exchangeable Runs

Randall D. Tobias, SAS Institute Inc.  
Box 8000, Cary, North Carolina, 27512.

## Abstract

Popular algorithms for optimal design do not account for non-exchangeability of the basic experimental units. This paper introduces an algorithm for searching for D-optimal experimental designs in the presence of certain kinds of non-exchangeability—in particular, when there are fixed covariates associated with units and/or when the units have a non-trivial covariance structure; this algorithm extends and improves on the one of Cook and Nachtsheim (1989) for finding optimal block designs. Examples are given where the influence of the covariates/covariance on the choice of design is known or at least roughly apparent, and the algorithm proves to be quite successful in these situations.

**Key words:** block designs, autocorrelation, factorial design, general linear model, neighbor-balanced designs, search algorithms

## 1 Introduction

Facilities for finding optimal designs have become standard components of software for computer-aided experimentation (Morris et al. 1992). They are useful when classical combinatorial techniques for experimental design do not apply—for example, if there is a non-standard design region or model. Given a candidate region of experimentation, common algorithms select the design by adding and deleting candidate points one at a time until a certain optimality criterion no longer improves (eg. Mitchell 1974a; see Cook and Nachtsheim 1980 for a survey). A fundamental assumption is that all the basic experimental units are completely *exchangeable*: given a set of factor combinations constituting a design, it makes no difference which combinations are applied to which units.

However, there are many experimental situations in which the basic experimental units are not exchangeable. Often they are grouped into *blocks*—days, machines, operators—so that units in the same block are exchangeable, but not units in different blocks. More generally, there may be known covariate values associated with each unit. More generally still (as we shall see), the units may have a non-trivial covariance structure—eg., a time-series correlation. It is difficult to apply standard optimal design algorithm in these cases.

Not that they haven't been tried. The second example in Mitchell (1974a), a seminal paper introducing the popular DETMAX algorithm for optimal design, involves a sampling plan over the course of a year, with a model incorporating time, measured in months, as a fixed effect. It is clear that a satisfactory plan will involve one sample per month, but there is no way to constrain DETMAX to find only such designs, and in fact some of the optimal designs which DETMAX produced did *not* sample all months. Moreover, the fixed month effect probably did not really model the effect of time correctly in this case, at least not if the intent was to make months which are closer together more related: for example, observations in the middle four months turn out to be independent of the time effect. A time-series covariance model would perhaps have been more appropriate, but DETMAX cannot take such a model into account.

Building on the work of Harville (1974), Cook and Nachtsheim (1989) (referred to here as CN) consider computer-aided optimal design for one common form of non-exchangeability, that of block structure. Given a set of block sizes and a candidate set of “treatments” (which may in fact be factor combinations) to be applied within blocks, their algorithm relies on efficient formulas for (a) exchanging a treatment in the design with one from the candidate set and (b) interchanging treatments in two different blocks. The performance of the algorithm is typical of techniques for computer-aided optimal design: while it reliably yields exact optimal designs in small-to-moderate cases, it invariably finds very efficient designs in all cases.

The first goal of this paper is to show how a modification of the CN algorithm (randomizing the choice of starting design) makes a significant improvement in the efficiency of the resulting designs. The second goal is to generalize the algorithm to find designs that are optimal for a much broader class of non-exchangeable experimental units—in particular, for arbitrary fixed covariates and/or correlated runs. Section 2 discusses the improvements and gives the more general form of the algorithm, and Section 3 gives several examples of its use. Comments on the examples will highlight the symmetries and structure of the resulting designs. Finally, Section 4 discusses the relation between algorithms like the one presented here and methods of design construction which focus more directly on symmetry and balance, and reflects on the proper use of optimal design search algorithms in the over-all context of experimentation.

## 2 Description of the Algorithm

### 2.1 The Setting

The setting for our discussion will be a general linear factorial model. For each of  $N$  collections of values  $\mathbf{f}$  for various factors a response  $y$  is to be observed. The expected value of  $y$  depends linearly on  $p$  different functions  $g_i(\mathbf{f})$  of the factor values and possibly on the values of known covariates  $\mathbf{z}$ :

$$E(y|\mathbf{f}) = \mathbf{g}(\mathbf{f})'\beta + \mathbf{z}'\theta. \tag{1}$$

We will call  $\mathbf{g}(\mathbf{f})$  the *treatment* or *design point* associated with the factor settings  $\mathbf{f}$ , and the  $\beta_i$  the *treatment effects*: because we are considering only finite candidate sets, we will ignore the dependence of the treatments on underlying factors and denote them by simply  $\mathbf{x} = \mathbf{g}(\mathbf{f})$  for the rest of this paper. The matrix  $X$  (respectively,  $Z$ ) whose rows consist of the treatments (resp. covariates) for the  $N$  factor settings (resp. units) which are actually used will be called the *treatment* (resp. *covariate*) *design matrix*. Then from (1) the expectation of the vector  $\mathbf{y}$  of observed responses is

$$E\mathbf{y} = X\beta + Z\theta. \quad (2)$$

Note that qualitative block effects can be modelled using 0-1 covariates indicating which block contains each unit. We denote the variance-covariance matrix of  $\mathbf{y}$  by  $V$ .

General criteria for designs are often defined in terms of the variance matrix of the estimated treatment effects  $\hat{\beta}$ , or equivalently in terms of its inverse, the *information matrix*, which has the following general form:

$$M = X' \left( V^{-1} - V^{-1}Z \left( Z'V^{-1}Z \right)^{-1} Z'V^{-1} \right) X. \quad (3)$$

When there are no covariates, the information matrix is just  $X'V^{-1}X$ , while if  $V$  is proportional to the identity (that is, if all units are uncorrelated with a common variance) then we can equivalently work with

$$M = X' \left( I - Z \left( Z'Z \right)^{-1} Z' \right) X. \quad (4)$$

Throughout this paper we will be concerned with optimizing the determinant of  $M$  (so-called *D-optimality*) but we will consider the trace of  $M^{-1}$  as a design criterion as well.

## 2.2 The Algorithm of Cook and Nachtsheim (1989)

The general structure of the CN algorithm is shown in figure 1. It begins with a candidate set  $C \subset \mathcal{R}^p$  of treatments and experimental units organized into  $b$  blocks, the  $i^{\text{th}}$  block having  $n_i$  plots, where  $\sum_i n_i = N$ . The design problem is to associate treatments with units—that is, to choose  $\mathbf{x}_{ij} \in C$  for  $j = 1, \dots, n_i$ ,  $i = 1, \dots, b$ —so as to maximize the determinant of the information matrix  $M$  for the treatment effects: CN consider the covariance structure of equation (4), where the columns of  $Z$  indicate which block contains the corresponding unit.

The CN algorithm works by successively improving a starting design, interchanging points in different blocks and exchanging them with points in the candidate set. The exchange step uses the *k-exchange* idea of Johnson and Nachtsheim (1983), where  $k$  is chosen beforehand: CN suggest using a value of  $k \leq N/4$ . Note, however, that if the size of  $C$  is much smaller than  $N$ , then the interchange step will dominate the run-time for the algorithm. In this case,  $k$  may as well be set to  $N$ ; the resulting algorithm for the exchange step is related to the modified Federov algorithm of Cook and Nachtsheim (1980).

### 2.3 General Update Formulas

The crucial parts of the CN algorithm are the evaluation of  $\Delta_E$  and  $\Delta_I$ , the multiplicative change in  $|M|$  for exchanges and interchanges, respectively. CN give efficient formulas for these based on rank-two updates for  $M$ : that is, after an exchange or an interchange, the updated form of the information matrix can be written as

$$M^* = M \pm AB' \quad (5)$$

for certain  $p \times 2$  matrices  $A$  and  $B$ . It follows that

$$(M^*)^{-1} = M^{-1} \mp M^{-1}A[I_2 \pm B'M^{-1}A]B'M^{-1} \text{ and} \quad (6)$$

$$|M^*| = |M| \times |I_2 \pm B'M^{-1}A|. \quad (7)$$

Thus, both  $\Delta_E(x_{ij}, x)$  and  $\Delta_I(x_{ij}, x_{kl})$  can be efficiently evaluated as the determinants of  $2 \times 2$  matrices.

The update formulas given by CN apply to block designs in particular. The following theorem provides rank-two update formulas for the general case of fixed covariates  $Z$  as discussed above, generalizing equations (10) and (13) of CN.

**Theorem 1** *Let  $M = X'(I - P)X$  where  $X' = \{\mathbf{x}_1 \dots \mathbf{x}_N\}$ .*

1. *Consider replacing the  $j^{\text{th}}$  row  $\mathbf{x}_j$  of  $X$  with  $\mathbf{x}$ . Let  $\mathbf{d}_j = \mathbf{x} - \mathbf{x}_j$  and let  $\mathbf{f}_j$  be the  $j^{\text{th}}$  row of  $X'P$ . Then the updated form of  $M$  is  $M + A_j B_j'$  where*

$$\begin{aligned} A_j &= \begin{bmatrix} \mathbf{x} - \mathbf{f}_j - (1 + P_{jj})\mathbf{d}_j & \mathbf{d}_j \\ & \mathbf{d}_j \end{bmatrix} \\ B_j &= \begin{bmatrix} & \mathbf{d}_j \\ & \mathbf{x} - \mathbf{f}_j \end{bmatrix}. \end{aligned} \quad (8)$$

2. *Consider interchanging the  $j^{\text{th}}$  and  $k^{\text{th}}$  rows of  $X$ . Let*

$$\begin{aligned} \mathbf{d}_{jk} &= \mathbf{x}_j - \mathbf{x}_k, \\ \mathbf{f}_{jk} &= \mathbf{f}_j - \mathbf{f}_k, \text{ and} \\ \nu &= P_{jj} + P_{kk} - 2(1 + P_{jk}). \end{aligned}$$

*Then the updated form of  $M$  is  $M - A_{jk} B_{jk}'$  where*

$$\begin{aligned} A_{jk} &= \begin{bmatrix} \mathbf{d}_{jk} - (\mathbf{f}_j - \mathbf{f}_k) & \mathbf{d}_{jk} \\ & \mathbf{d}_{jk} \end{bmatrix} \\ B_{jk} &= \begin{bmatrix} & \mathbf{d}_{jk} \\ & (1 + \nu)\mathbf{d}_{jk} - (\mathbf{f}_j - \mathbf{f}_k) \end{bmatrix} \end{aligned} \quad (9)$$

When there are covariates  $Z$ ,  $P$  in the theorem has the form  $I - Z(Z'Z)^{-1}Z'$  (see equation 4 above). Since  $P$  doesn't change, the most efficient way to implement the update formulas is to compute it once at the beginning of the search. If the design is very large this may not be possible: in special cases, such as block designs,  $X'P$  and the individual  $P_{jk}$  can be efficiently computed without forming  $P$  itself.

On the other hand, note that the update formulas in the theorem only depend on  $Z$  through  $P$ ; they apply no matter how  $P$  is formed. Thus, if we have a covariance matrix  $V$  for the runs, the theorem can be applied with  $P = I - V^{-1}$  to provide update formulas for  $X'V^{-1}X$ , the information matrix of the treatment effects in this case.

## 2.4 Initial Designs

The starting point for any optimization technique can greatly affect the quality of the result, and this is no less true for optimal design algorithms: for example, Galil and Kiefer (1980) show how the efficiency of standard optimal design algorithms may be greatly improved by judicious choice of the initial design. In particular, because the algorithms are invariably subject to becoming “trapped” at local optima which are not globally optimal, a certain amount of randomness in the starting design is desirable; typically, the search is run a number of times from different random starting designs.

The CN algorithm as originally stated requires that  $M$  be non-singular, and CN give a method of obtaining an initial block design which always is (see figure 1). However, this initial design does not include any of the randomness which is so useful in overcoming the problem of local optima; moreover, CN's initialization method is not applicable to situations involving more general covariates than simple blocks. In fact, in the general case it might be difficult to generate a non-singular design at all. Mitchell's (1974a) approach to the problem of a singular information matrix is to start the search with  $X'X + \alpha I$  instead of  $X'X$ , where  $\alpha$  is a small, positive number: the perturbed matrix is always positive definite and if the perturbation is small the optimum design is unlikely to be any different.

Mitchell's perturbation idea can be applied to the case of designs with fixed covariates or covariance, too. Accordingly, we initialize the search with a *randomly* selected set of treatments and optimize  $X'(I - P)X + \alpha I$ . It might be expected that working with this perturbed version of the information matrix for the treatment effects will affect how well the search succeeds at finding the *exact* optimum in some cases; a reasonable procedure might be to revert to  $X'(I - P)X$  when it becomes non-singular.

## 2.5 COVMAX: A General Algorithm for Designs with Covariance

Figure 2 gives a general algorithm which we call COVMAX, for optimizing  $|X'(I - P)X|$ , incorporating the update formulas of section 2.3 as well as the recommendations of section 2.4 for initialization. COVMAX is clearly patterned after the block design algorithm of CN (figure 1), but the new initialization method in conjunction with the idea of optimizing a perturbed information matrix provide great improvements over CN in terms of the ability of the algorithm to find globally optimal designs. Moreover, COVMAX applies much more generally than CN. The examples in the next section illustrate these points.

The run-time efficiency of COVMAX is basically the same as CN, depending primarily on how often and how efficiently the exchange and interchange criteria  $\Delta_E$  and  $\Delta_I$  are computed.

## 2.6 Efficiency Measures

The determinant of  $M$  is the  $p^{\text{th}}$  power of the geometric mean of the variances of the canonical estimable functions in the linear model. This interpretation may give the unscaled value of  $|M|^{1/p}$  an intrinsic meaning, but it is usually more useful to compare it to a reference value for a good (but perhaps hypothetical) design. For example, Mitchell (1974b) compares  $|M|^{1/p}$  to  $N$  for two-level screening designs, since  $N$  is the value of  $|M|^{1/p}$  for orthogonal arrays with columns coded as  $\pm 1$ . The same approach is useful in the general case if the candidate points are coded so that  $X'_C X_C = N_C I$ , where  $X_C$  is the design matrix over all the candidates and  $N_C$  is the number of candidates. In this case, reasonable measures of the D- and A-efficiencies of a design in the presence of fixed covariates are given by  $|M|^{1/p}/N$  and  $(p/N)/\text{trace}(M^{-1})$ , respectively. More generally, when  $M = X'AX$  we compute efficiencies as follows:

$$\begin{aligned} \text{D-efficiency} &= c_D^{-1} \cdot |M|^{1/p}/N, & c_D &= \prod_{i=1}^p \lambda_i^{1/p}, \\ \text{A-efficiency} &= c_A \cdot (p/N)/\text{trace}(M^{-1}) & c_A &= \sum_{i=1}^p \lambda_i/p, \end{aligned} \tag{10}$$

where  $\lambda_1, \dots, \lambda_p$  are the  $p$  largest eigenvalues of  $A$ .

## 3 Examples

The COVMAX algorithm presented in figure 2 has been experimentally implemented in the C programming language and tested on a HP-700 workstation. The implementation provides for general linear modelling for both the treatment and covariate models, with quantitative and qualitative effects as well as any degree of interaction. It is also possible to input a matrix as a general covariance structure  $V$  for the runs. The flexibility and efficiency of the implementation, as well as the speed of the HP-700 processor, made it possible to test the algorithm in a wide variety of situations. In

this section we report on some of them. All three examples involve simple qualitative treatments, since that is the easiest case for exploring the characteristics of the final design. However, neither the algorithm nor the implementation are limited to such a treatment structure.

### 3.1 Incomplete Block Designs

CN claim good success using their algorithm to find traditional incomplete block designs for qualitative treatments. In particular, concerning its capacity to find balanced incomplete block designs (BIBD's) they remark, "In our experience, the algorithm has not failed to find such designs when they are known to exist." This is very encouraging: while BIBD's are among the most widely studied combinatorial objects, there are still many open existence problems. A method which always finds a BIBD if it exists would be a useful tool indeed.

Unfortunately, the CN algorithm is not perfect. It does indeed succeed at finding relatively small balanced designs when they exist; but as the number of treatments and blocks increases, the algorithm finds the balanced design less often, although it will invariably find a very D-efficient one. A catalogue of BIBD's was assembled from various sources (primarily Mathon and Rosa, 1985); tables 1-3 report the results of using various algorithms to find designs in 200 of the smallest cases for which balanced designs are known to exist. For each possible BIBD for  $t$  treatments in  $b$  blocks of size  $k$ , the negative log deficiencies for the results of three algorithms are shown:  $D_0$  corresponds to the result of CN's original algorithm, using their chaining method of constructing the initial design;  $D_R$  corresponds to COVMAX, using ten different random starting designs; and  $D_S$  corresponds to using a standard D-optimal search algorithm (the "modified Federov" algorithm of Cook and Nachtsheim (1980)) with the full set of  $tb$  combinations of treatments and blocks as candidate. (There is no guarantee that this third method will result in equi-sized blocks, but in fact it did for all the cases shown here.) The deficiency for a design is defined as one minus the standard D-efficiency for block designs, which is the ratio of  $|M|^{1/(t-1)}$  to the value for this quantity in a balanced design. Thus, the negative log deficiency is infinite for balanced designs (printed in the tables as "\*") and larger values correspond to better designs: for reference, note that negative log deficiency values of 7, 8, 9, and 10 correspond to efficiency factors of 99.9%, 99.97%, 99.99%, 99.995%, and 99.998%, respectively.

The first thing to note is that all three algorithms give very efficient designs in all cases, even the standard optimal design search procedure which does not fix the block structure. If the goal is just to obtain a good design, there is really not much practical difference between the three approaches. However, if the object is to find a balanced design, it's apparent that the standard search does considerably poorer: it finds the BIBD less than 10% of the time, while the CN method finds it nearly half and COVMAX almost two-thirds of the time. The randomness in the new method improves the partial results, too: the efficiency of the design found by this method is nearly always better than that found by CN. For the remainder of the paper application of COVMAX will mean

using the algorithm in figure 2 some number (usually 10) of times from different random starting designs.

Although the new method is better than CN and considerably better than simply searching among all block×treatment pairs, it can still reliably generate balanced designs only in relatively small cases. It is nonetheless tempting to try the algorithm in a case for which the existence of a balanced design is not yet determined. The smallest such case has  $t = 46$ ,  $b = 69$ , and  $k = 6$ . In 100 attempts using COVMAX, starting from different random initial designs, the best D-efficiency achieved was only about 99.82%. As noted above, however, this is not conclusive evidence that a balanced design does not exist—simply that if it does, COVMAX was not able to find it. Is COVMAX usually able to find balanced designs in this range? Figure 3 plots run-time×D-efficiency for searches with this set of parameters as well as two others for which BIBD’s are known to exist: the indication is that the behaviour of COVMAX in this case is not inconsistent with its behaviour when a design does in fact exist. Thus, we can say nothing for certain about the existence or non-existence of a balanced design for this set of parameters.

### 3.2 Designs for Qualitative Treatments with Covariates

Harville (1975) considers optimal designs for qualitative treatment effects in the presence of a single quantitative covariate. He shows that optimality is essentially equivalent to having the mean covariate values for the treatment groups as similar as possible. Harville presents a 10-run example, with covariate values given by

$$\{ .46 .54 .58 .60 .73 .77 .82 .84 .89 .95 \},$$

with from 2 to 6 treatments and with both a linear and a quadratic covariate model. COVMAX confirms the D-optimality of Harville’s designs in this case: Table 4 presents the D- and A-efficiencies of Harville’s designs, as calculated using (10) above. Note that the two-treatment design with a linear covariate model has 100% efficiency: in this case, the mean covariate values in the two treatment groups match exactly.

More generally, it is not difficult to see that an optimal design for qualitative treatments with an  $m^{\text{th}}$ -order polynomial model for a single covariate will have the first  $m$  moments of the distributions of the covariate within each treatment group as similar as possible. For example, 1000 random variates were generated, distributed as a uniform random mixture of two normals,  $N(0,1)$  and  $N(10,3)$ , with a mixing parameter of 0.65:

$$Y = \chi(U < 0.65)\epsilon_1 + \chi(U \geq 0.65)\epsilon_2, \quad U \sim U(0, 1), \quad \epsilon_1 \sim N(0, 1), \quad \epsilon_2 \sim N(10, 3).$$

COVMAX was used to search for an optimal design for five treatments with a fourth-order polynomial model in  $Y$ : the best design had a D-efficiency (as calculated using equation 10 above) of

99.9997; table 5 and figure 4 confirm that the distributions of  $Y$  within each treatment group are very similar.

Designs like this may also be useful in statistical resampling applications, when it is necessary to sample without replacement while making sure that the distributions of individual subsamples are as similar as possible.

### 3.3 Experiments with Spatial Correlation

Covariate effects are included in designs in order to account for the notion that experimental units which are more closely related are more likely to have similar responses, whatever treatment is applied. When there is a reasonable measure of the “distance” between experimental units, another way of accounting for their innate relationship is to postulate a *covariance* between units which is a function of their distance. For example, if observations are made over time, an autoregressive covariance may be appropriate; as another alternative, a spatial correlation structure can be used if experimental units are plots of land in an agricultural field trial. See Jennrich and Schluter (1986), Cressie (1991), and the chapter on the MIXED procedure in SAS Institute (1992) for more information about the analysis of experiments with non-trivial covariance structures.

COVMAX can be used to find designs optimal for a specific covariance. Moreover, a design which is optimal for one particular member of a given general class is often likely to be efficient for most members of that class. For example, optimal designs for autoregressive covariance structures will probably require first-order neighbor balance; conversely, any neighbor-balanced design is likely to be efficient for a wide variety of time-related covariances.

To illustrate this point, consider a design for  $t = 5$  qualitative treatments in  $N = 11$  runs having an autocorrelated covariance:

$$V_{ij} = \text{cov}(Y_i, Y_j) = \sigma^2 \rho^{|i-j|}, \quad i, j = 1, \dots, N. \quad (11)$$

Since the optimal design doesn't depend on  $\sigma^2$ , we use  $\sigma^2 = 1$  for simplicity. Setting  $\rho = 0.9$  and applying COVMAX yielded an optimal design which put the treatments in the following order:

4 2 5 3 1 2 3 4 1 5 4

This design does indeed exhibit first-order neighbor balance: every pair of treatments occur as neighbors exactly once. Moreover, this design, generated for a particular value of  $\rho$ , is also apparently optimal for all  $\rho \in [0, 1)$ .

Neighbor-balanced sequences are equivalent to traversals of certain graphs (Hutchinson and Wilf, 1975). Using this representation, it can be shown that a neighbor-balanced design must have at

least  $t(t-1)/2 + 1$  runs for  $t$  odd and at least  $t(t-1) + 1$  runs for  $t$  even, and that such minimum designs exist for all  $t$ . COVMAX easily finds such designs for  $t \leq 20$  or so.

Designs with various kinds of neighbor balance have been widely studied, but usually the object is to superimpose neighbor balance on an already existing structure—to find BIBD’s which are also neighbor balanced, for example. COVMAX can be used to find such designs also, by combining covariates  $Z$  with covariance. In this case, the information matrix for the treatments is given by equation (3). Clearly, the general algorithm applies, using  $\left(V^{-1} - V^{-1}Z(Z'V^{-1}Z)^{-1}Z'V^{-1}\right)^{-1}$  as the covariance matrix for the units. Using this technique, COVMAX can be used to find neighbor-balanced BIBD and Latin square designs of moderate order ( $t \leq 11$ ). For example, table 5 gives a balanced incomplete block design for 7 treatments in 7 blocks of size 4 found by COVMAX which also exhibits first-order neighbor balance.

## 4 “Optimal” Design Versus Combinatorial Design

We have presented a new algorithm, COVMAX, for searching for D-optimal experimental designs with non-exchangeable runs; and we have shown how this algorithm both generalizes and improves upon the one of CN for optimal block designs in particular. Incidentally, COVMAX also finds somewhat better designs than the algorithm of CN for some of the examples mentioned in that paper itself, though the improvement is at most a few percentage points of the efficiencies as measured by (10). For example, for the two-factor response surface design in two blocks of size 8 and 6, respectively, the design which COVMAX finds is about 1% better and slightly more symmetrical: it can be derived from CN’s (9,5) design by moving one centerpoint from the larger block to the smaller one. On the other hand, while the COVMAX design for the three-factor response surface problem in 5 blocks of size 4 is about 5% better, it doesn’t seem to have the “near-symmetry” which CN find in their design.

The use of designs constructed using D-optimal search algorithms is somewhat controversial. Both their enthusiasts and their detractors see them as supplanting traditional approaches to design construction, which emphasize combinatorial ideas like balance and orthogonality. There are at least three important reasons why “search optimal” designs do not make combinatorial design theory obsolete.

1. Optimal design search algorithms are imperfect and specialized. As the example in section 3.1 shows, combinatorial methods sometimes provide better designs than the search algorithms according to the criterion of search itself. Moreover, the designs generated by search algorithms are only guaranteed to be optimal for at most one criterion. Combinatorial methods, by contrast, provide designs which are often optimal in many different ways.

2. The quality of an experiment may actually improve if the researcher is required to force it into a form where a combinatorial design will apply. For example, while a researcher may initially conceive an experiment to involve many levels for the factors, it is usually better to whittle these down to just two; while in principle an optimal design can be generated for any combination of numbers of factor levels, orthogonal two-level experiments are usually preferable in order to screen for the factors which really count.
3. There will always be situations involving specialized constraints which call for specialized construction methods. The COVMAX algorithm addresses an important class of constraints, but there are others: situations that come to mind include problems in which covariate and treatment effects interact, or when different treatments have different costs.

The crucial point is that “good” designs have high degrees of symmetry. Information-based criteria work because their optimization tends to be associated with such structure; combinatorial methods of design construction focus on symmetry directly. For example, balance in incomplete block designs was not originally conceived of in order to provide an information matrix which satisfies the hypothesis of Kiefer’s General Equivalence Theorem, but because of the intuition that the design shouldn’t distinguish between treatments for the purposes of analysis.

There are some general cases where it is probably safe enough to use an optimal design search algorithm relatively blindly—eg. unbalanced designs for two-level factors, or designs for constrained mixture experiments. But it is the opinion of this author that the first thing to do with the product of an optimal design search algorithm in a novel set-up is to explore its symmetries thoroughly to see whether they accurately reflect the researchers intuition about the experimental situation. If they do not, we probably need to go back and re-think the set-up. In this vein, the algorithm presented in this paper makes it possible to impose a certain amount of structure or symmetry on the design. As with all optimal design tools, it really becomes useful when coupled with versatile facilities for exploring the resulting designs.

## 5 References

- Cook, R.D. and Nachtsheim, C.J. (1980). “A Comparison of Algorithms for Constructing Exact D-optimal Designs.” *Technometrics* 22, pp. 315-324.
- Cook, R.D. and Nachtsheim, C.J. (1989). “Computer-Aided Blocking of Factorial and Response-Surface Designs.” *Technometrics* 31, pp. 339-346.
- Cressie, N. (1991). *Statistics for Spatial Data*. New York: John Wiley and Sons.

- Galil, Z. and Kiefer, J. (1980). "Time- and Space-saving Computer Methods, Related to Mitchell's DETMAX, for Finding D-Optimum Designs." *Technometrics* 22, pp. 301-313.
- Harville, D. (1974). "Nearly Optimal Allocation of Experimental Units Using Observed Covariate Values." *Technometrics* 16, pp. 589-599.
- Hutchinson, J.P. and Wilf, H.S. (1975). "On Eulerian Circuits and Words with Prescribed Adjacency Patterns." *Journal of Combinatorial Theory, Series A* 18: pp. 80-87.
- Jennrich, R.I. and Schlucter, M.D. (1986). "Unbalanced Repeated Measures Models with Structured Covariance Matrices." *Biometrics* 42, pp. 805-820.
- Johnson, M.E. and Nachtsheim, C.J. (1983). "Some Guidelines for Constructing Exact D-optimal Designs on Convex Design Spaces." *Technometrics* 25, pp. 271-277.
- Mardia, K.V., Kent, J.T., and Bibby, J.M. (1979). *Multivariate Analysis*. Academic Press: New York.
- Mathon, R. and Rosa, A. (1985). "Tables of Parameters of BIBDs with  $r \leq 41$  including Existence, Enumeration, and Resolvability Results." *Annals of Discrete Mathematics* 26: pp. 275-308.
- Mitchell, T.J. (1974a). "An Algorithm for the Construction of D-optimal Experimental Designs." *Technometrics* 20, pp. 203-210.
- Mitchell, T.J. (1974b). "Computer Construction of 'D-optimal' First-Order Designs". *Technometrics* 20, pp. 211-220.
- Morris, W.T., Nachtsheim, C.J., and Welch, W. (1992). "Computer Software for the Design of Experiments." Presentation given at the 36<sup>th</sup> Annual Fall Technical Conference of the American Society for Quality Control, Philadelphia, PA.
- SAS Institute Inc. (1992). *SAS/STAT<sup>®</sup> Software: Changes and Enhancements, Release 6.07*, SAS<sup>®</sup> Technical Report P-229. Cary, NC: SAS Institute Inc.
- Tobias, R.D. (1986) *The Algebra of a Multistratum Design and the Application of its Structure to Analysis*. University of North Carolina Department of Statistics Mimeo Series #1599: Chapel Hill, NC.

## A Calculation of the Prediction Variance for Partitioned $X$

From formula (A.2.4g) of Mardia, Kent, and Bibby (1979) we have

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} = \begin{bmatrix} (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & - (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} A_{12}A_{22}^{-1} \\ -A_{22}^{-1}A_{21} (A_{11} - A_{12}A_{22}^{-1}A_{21})^{-1} & (A_{22} - A_{21}A_{11}^{-1}A_{12})^{-1} \end{bmatrix} \quad (12)$$

Thus, if  $X = (X_1 \ X_2)$  then from (12) we have

$$\begin{aligned} (X'X)^{-1} &= \begin{bmatrix} X_1'X_1 & X_1'X_2 \\ X_2'X_1 & X_2'X_2 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} (1,1) & (1,2) \\ (2,1) & (2,2) \end{bmatrix}^{-1} \end{aligned} \quad (13)$$

where

$$\begin{aligned} (1,1) &= (X_1'X_1 - X_1'X_2(X_2'X_2)^{-1}X_2'X_1)^{-1} \\ (1,2) &= - (X_1'X_1 - X_1'X_2(X_2'X_2)^{-1}X_2'X_1)^{-1} X_1'X_2(X_2'X_2)^{-1} \\ (2,1) &= - (X_2'X_2)^{-1}X_2'X_1 (X_1'X_1 - X_1'X_2(X_2'X_2)^{-1}X_2'X_1)^{-1} \\ (2,2) &= (X_2'X_2 - X_2'X_1(X_1'X_1)^{-1}X_1'X_2)^{-1} \end{aligned} \quad (14)$$

Now, let

$$\begin{aligned} B_{ij} &= (X_i'X_i)^{-1}X_i'X_j \\ P_{ii} &= X_i(X_i'X_i)^{-1}X_i' \end{aligned} \quad (15)$$

Then we can write (13) and (14) as

$$(X'X)^{-1} = \begin{bmatrix} (X_1'(I - P_{22})X_1)^{-1} & - (X_1'(I - P_{22})X_1)^{-1} B'_{21} \\ -B_{21} (X_1'(I - P_{22})X_1)^{-1} & (X_2'(I - P_{11})X_2)^{-1} \end{bmatrix} \quad (16)$$

Further, let

$$\begin{aligned} Q_{ii} &= I - P_{ii} \\ M_{11} &= X_1'Q_{22}X_1 \\ M_{22} &= X_2'Q_{11}X_2 \end{aligned} \quad (17)$$

Notice that for Cook and Nachstheim,  $Q_2 = Q_{22}$  and  $M = M_{11}$ . Then (16) becomes

$$(X'X)^{-1} = \begin{bmatrix} M_{11}^{-1} & -M_{11}^{-1}B'_{21} \\ -B_{21}M_{11}^{-1} & M_{22}^{-1} \end{bmatrix} \quad (18)$$

Now, suppose  $x$  is partitioned similarly to  $X$ , with  $x^1 = [x_1^1 \ x_2^1]'$  and  $x_2 = [x_1^2 \ x_2^2]'$ . Then

from (18) we have

$$\begin{aligned}
\begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix}' (X'X)^{-1} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} &= x_1^{1'} M_{11}^{-1} x_1^2 - x_1^{1'} M_{11}^{-1} B'_{21} x_2^2 - x_2^{1'} B_{21} M_{11}^{-1} x_1^2 + x_2^{1'} M_{22}^{-1} x_2^2 \\
&= x_2^{1'} M_{22}^{-1} x_2^2 + (x_1^1 - B'_{21} x_2^1)' M_{11}^{-1} (x_1^2 - B'_{21} x_2^2) - x_2^{1'} B_{21} M_{11}^{-1} B'_{21} x_2^2 \\
&= x_2^{1'} (M_{22}^{-1} - B_{21} M_{11}^{-1} B'_{21}) x_2^2 + (x_1^1 - B'_{21} x_2^1)' M_{11}^{-1} (x_1^2 - B'_{21} x_2^2)
\end{aligned} \tag{19}$$

Similarly

$$X (X'X)^{-1} X' = X_2 (M_{22}^{-1} - B_{21} M_{11}^{-1} B'_{21}) X_2' + (X_1' - B'_{21} X_2')' M_{11}^{-1} (X_1' - B'_{21} X_2'). \tag{20}$$

But

$$\begin{aligned}
X_1' - B'_{21} X_2' &= X_1' - X_1' X_2 (X_2' X_2)^{-1} X_2' \\
&= X_1' - X_1' P_{22} \\
&= X_1' Q_{22}
\end{aligned}$$

so that

$$\begin{aligned}
(X_1' - B'_{21} X_2')' M_{11}^{-1} (X_1' - B'_{21} X_2') &= Q_{22} X_1 (X_1' Q_{22} X_1)^{-1} X_1' Q_{22} \\
&= \text{Proj}(\mathcal{R}(Q_{22} X_1)).
\end{aligned} \tag{21}$$

But  $X (X'X)^{-1} X' = \text{Proj}(\mathcal{R}(X_1) + \mathcal{R}(X_2))$ , so that (20) and (21) imply

$$\begin{aligned}
X_2 (M_{22}^{-1} - B_{21} M_{11}^{-1} B'_{21}) X_2' &= \text{Proj}(\mathcal{R}(X_1) + \mathcal{R}(X_2)) - \text{Proj}(\mathcal{R}(Q_{22} X_1)) \\
&= \text{Proj}(\mathcal{R}(X_2)) = P_{22}
\end{aligned} \tag{22}$$

by A1.7 of Tobias (1986). Then if  $x_2^{1'} = a' X_2$ ,  $x_2^{2'} = b' X_2$  (if, for example, they're both rows of  $X_2$ ) we have

$$\begin{aligned}
\begin{bmatrix} x_1^1 \\ x_2^1 \end{bmatrix}' (X'X)^{-1} \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} &= a' X_2 (M_{22}^{-1} - B_{21} M_{11}^{-1} B'_{21}) X_2' b + (x_1^1 - B'_{21} x_2^1)' M_{11}^{-1} (x_1^2 - B'_{21} x_2^2) \\
&= a' X_2 (X_2' X_2)^{-1} X_2' b + (x_1^1 - B'_{21} x_2^1)' M_{11}^{-1} (x_1^2 - B'_{21} x_2^2) \\
&= x_2^{1'} (X_2' X_2)^{-1} x_2^2 + (x_1^1 - B'_{21} x_2^1)' M_{11}^{-1} (x_1^2 - B'_{21} x_2^2).
\end{aligned}$$

The upshot of all this is that computations that rely on the prediction variance (such as D-optimal design searching) can work with  $B_{21}$  and  $M_{11}$  if  $X_2$  is fixed, rather than the potentially much larger  $X$  and  $X'X$ .

## B Updating $M_{11}$

### B.1 Exchanges

Recall that

$$\begin{aligned}
M_{11} &= X_1' (I - X_2 (X_2' X_2)^{-1} X_2') X_1 \\
&= X_1' X_1 - X_1' X_2 (X_2' X_2)^{-1} X_2' X_1
\end{aligned}$$

What happens to  $M_{11}$  when we change the  $j^{\text{th}}$  row of  $X_1$ ? We can see that it's only affected through

$$\begin{aligned} X_1' X_1 &= \sum_j x_{1j} x_{1j}' \text{ and} \\ X_2' X_1 &= \sum_j x_{2j} x_{1j}'. \end{aligned}$$

If we change  $x_{1j}'$  to  $x$  then

$$\begin{aligned} (X_1' X_1)^* &= X_1' X_1 - x_{1j} x_{1j}' + x x' \text{ and} \\ (X_2' X_1)^* &= X_2' X_1 - x_{2j} x_{1j}' + x_{2j} x' \\ &= X_2' X_1 - x_{2j} (x_{1j} - x)' \\ &= X_2' X_1 + x_{2j} d_j' \end{aligned} \tag{23}$$

where  $d_j = x - x_{1j}$ , as in Cook and Nachsheim (1989). Then

$$\begin{aligned} M_{11}^* &= (X_1' X_1)^* - (X_2' X_1)^* (X_2' X_2)^{-1} (X_2' X_1)^* \\ &= X_1' X_1 - x_{1j} x_{1j}' + x x' - (X_1' X_2 + d_j x_{2j}') (X_2' X_2)^{-1} (X_2' X_1 + x_{2j} d_j') \\ &= X_1' X_1 - x_{1j} x_{1j}' + x x' \\ &\quad - X_1' X_2 (X_2' X_2)^{-1} X_2' X_1 - d_j x_{2j}' (X_2' X_2)^{-1} X_2' X_1 \\ &\quad - X_1' X_2 (X_2' X_2)^{-1} x_{2j} d_j' - d_j x_{2j}' (X_2' X_2)^{-1} x_{2j} d_j' \\ &= M_{11} - x_{1j} x_{1j}' + x x' - d_j x_{2j}' B_{21} - B_{21}' x_{2j} d_j' - z_j d_j d_j' \\ &= M_{11} - x_{1j} x_{1j}' + x x' - d_j f_j' - f_j d_j' - z_j d_j d_j' \end{aligned}$$

where

$$\begin{aligned} z_j &= x_{2j}' (X_2' X_2)^{-1} x_{2j} \\ f_j &= B_{21}' x_{2j}. \end{aligned}$$

Note that for

$$\begin{aligned} A_j &= \begin{bmatrix} x - f_j - (1 + z_j) d_j & d_j \\ & & & & & \end{bmatrix} \\ B_j &= \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \end{aligned}$$

we have

$$A_j B_j' = x d_j - f_j d_j' - z_j d_j d_j' - d_j d_j' + d_j x' - d_j f_j'$$

so that

$$-d_j f_j' - f_j d_j' - z_j d_j d_j' = A_j B_j' - x d_j' - d_j x' + d_j d_j'$$

Then

$$\begin{aligned} M_{11}^* &= M_{11} - x_{1j} x_{1j}' + x x' + A_j B_j' - x d_j' - d_j x' + d_j d_j' \\ &= M_{11} - x_{1j} x_{1j}' + (x - d_j)(x - d_j)' + A_j B_j' \\ &= M_{11} - x_{1j} x_{1j}' + x_{1j} x_{1j}' + A_j B_j' \\ &= M_{11} + A_j B_j' \end{aligned} \tag{24}$$

Thus, when the  $j^{\text{th}}$  row of  $X_1$  is exchanged,  $M_{11}$  can be updated with a rank-2 update by means of the two  $p \times 2$  matrices  $A_j$  and  $B_j$ .

## B.2 Interchanges

Now suppose instead of changing  $x_{1j}$  with an arbitrary  $x$ , we interchange it with  $x_{1k}$ —that is, we switch

$$\begin{pmatrix} x_{1j} & x_{2j} \\ x_{1k} & x_{2k} \end{pmatrix} \rightarrow \begin{pmatrix} x_{1k} & x_{2j} \\ x_{1j} & x_{2k} \end{pmatrix}$$

Since  $X_2$  is still unchanged,  $M_{11}$  is again affected only through  $X_1'X_1$  and  $X_2'X_1$ . But since we are interchanging two rows of  $X_1$ ,  $X_1'X_1$  is also unchanged, while

$$\begin{aligned} (X_2'X_1)^+ &= X_2'X_1 - x_{2j}x'_{1j} + x_{2j}x'_{1k} - x_{2k}x'_{1k} + x_{2k}x'_{1j} \\ &= X_2'X_1 - (x_{2j} - x_{2k})(x_{1j} - x_{1k})' \\ &= X_2'X_1 - d_{2jk}d'_{1jk} \end{aligned}$$

where

$$\begin{aligned} d_{1jk} &= x_{1j} - x_{1k} \\ d_{2jk} &= x_{2j} - x_{2k} \end{aligned}$$

Then

$$\begin{aligned} M_{11}^+ &= X_1'X_1 - (X_1'X_2 - d_{1jk}d'_{2jk})(X_2'X_2)^{-1}(X_2'X_1 - d_{2jk}d'_{1jk}) \\ &= X_1'X_1 - X_1'X_2(X_2'X_2)^{-1}X_2'X_1 \\ &\quad + d_{1jk}d'_{2jk}(X_2'X_2)^{-1}X_2'X_1 \\ &\quad + X_1'X_2(X_2'X_2)^{-1}d_{2jk}d'_{1jk} \\ &\quad - d_{1jk}d'_{2jk}(X_2'X_2)^{-1}d_{2jk}d'_{1jk} \\ &= M_{11} + d_{1jk}d'_{2jk}B_{21} + B'_{21}d_{2jk}d'_{1jk} - z_{jk}d_{1jk}d'_{1jk} \\ &= M_{11} + d_{1jk}f'_{jk} + f_{jk}d'_{1jk} - z_{jk}d_{1jk}d'_{1jk} \end{aligned}$$

where

$$\begin{aligned} z_{jk} &= d'_{2jk}(X_2'X_2)^{-1}d_{2jk} \\ f_{jk} &= B'_{21}d_{2jk}. \end{aligned}$$

Now, let

$$\begin{aligned} A_{jk} &= \begin{bmatrix} d_{1jk} - f_{jk} & d_{1jk} \\ d_{1jk} & d_{1jk} - f_{jk} + \nu d_{1jk} \end{bmatrix} \\ B_{jk} &= \begin{bmatrix} d_{1jk} & d_{1jk} - f_{jk} + \nu d_{1jk} \end{bmatrix} \end{aligned}$$

Then

$$\begin{aligned} A_{jk}B'_{jk} &= d_{1jk}d'_{1jk} - f_{jk}d'_{1jk} + d_{1jk}d'_{1jk} - d_{1jk}f'_{jk} + \nu d_{1jk}d'_{1jk} \\ &= (\nu + 2)d_{1jk}d'_{1jk} - f_{jk}d'_{1jk} - d_{1jk}f'_{jk} \\ \Rightarrow -A_{jk}B'_{jk} &= f_{jk}d'_{1jk} + d_{1jk}f'_{jk} - (\nu + 2)d_{1jk}d'_{1jk} \end{aligned}$$

so that if we set

$$\begin{aligned} \nu + 2 &= z_{jk} \\ \Rightarrow \nu &= z_{jk} - 2 \end{aligned}$$

then we have

$$d_{1jk}f'_{jk} + f_{jk}d'_{1jk} - z_{jk}d_{1jk}d'_{1jk} = -A_{jk}B'_{jk}$$



Then combining (24) with (26) and (27) we have the following formula for the update to  $|X'X|$  when we change the  $j^{\text{th}}$  row of  $X_1$  to  $x$ :

$$\begin{aligned} |(X'X)^*| &= |X_2'X_2| \times |M_{11}^*| \\ &= |X_2'X_2| \times |M_{11} + A_j B_j'| \\ &= |X_2'X_2| \times |M_{11}| \times |I + B_j' M_{11}^{-1} A_j| \end{aligned} \quad (28)$$

where

$$\begin{aligned} A_j &= \begin{bmatrix} x - f_j - (1 + z_j)d_j & d_j \\ & x - f_j \end{bmatrix} \\ B_j &= \begin{bmatrix} & d_j \\ & x - f_j \end{bmatrix} \end{aligned} \quad (29)$$

with

$$\begin{aligned} d_j &= x - x_{1j} \\ z_j &= x'_{2j}(X_2'X_2)^{-1}x_{2j} \\ f_j &= B_{21}'x_{2j}. \end{aligned}$$

Note that  $f_k$  can be updated using

$$\begin{aligned} (X_1'X_2)^*(X_2'X_2)^{-1}x_{2k} &= (X_1'X_2 + d_j x'_{2j})(X_2'X_2)^{-1}x_{2k} \\ &= f_k + d_j x'_{2j}(X_2'X_2)^{-1}x_{2k} \\ &= f_k + [X_2(X_2'X_2)^{-1}X_2']_{jk} d_j. \end{aligned}$$

Likewise, combining (25) with (26) and (27) we have the following formula for the update to  $|X'X|$  when we interchange the  $j^{\text{th}}$  row of  $X_1$  with the  $k^{\text{th}}$ :

$$\begin{aligned} |(X'X)^*| &= |X_2'X_2| \times |M_{11}^*| \\ &= |X_2'X_2| \times |M_{11} - A_{jk} B_{jk}'| \\ &= |X_2'X_2| \times |M_{11}| \times |I - B_{jk}' M_{11}^{-1} A_{jk}| \end{aligned} \quad (30)$$

where

$$\begin{aligned} A_{jk} &= \begin{bmatrix} d_{1jk} - (f_j - f_k) & d_{1jk} \\ & A_{jk,1} + \nu d_{1jk} \end{bmatrix} \\ B_{jk} &= \begin{bmatrix} & d_{1jk} \\ & A_{jk,1} + \nu d_{1jk} \end{bmatrix} \end{aligned} \quad (31)$$

with

$$\begin{aligned} d_{1jk} &= x_{1j} - x_{1k} \\ \nu &= (z_j - 1) + (z_k - 1) - 2x'_{2j}(X_2'X_2)^{-1}x_{2k}. \end{aligned}$$

Here the update to  $f_l$  is given by

$$\begin{aligned} (X_1'X_2)^+(X_2'X_2)^{-1}x_{2l} &= (X_1'X_2 - d_{1jk}x'_{2j} + d_{1jk}x'_{2k})(X_2'X_2)^{-1}x_{2l} \\ &= f_k - d_{1jk}x'_{2j}(X_2'X_2)^{-1}x_{2l} + d_{1jk}x'_{2k}(X_2'X_2)^{-1}x_{2l} \\ &= f_k + \left( [X_2(X_2'X_2)^{-1}X_2']_{kl} - [X_2(X_2'X_2)^{-1}X_2']_{jl} \right) d_{1jk} \end{aligned}$$

## D Design for Fixed Covariance

Consider now updating  $X'VX$  and  $|X'VX|$  for given  $V$ ; the algebra is much the same as in B.2 and B.1. First, let  $R$  be a square root of  $V$ —for example, the Cholesky root—so that

$$X'VX = X'R'RX.$$

Then if the  $j^{\text{th}}$  row of  $X$  is changed to  $x$ , we have, similarly to 23

$$(RX)^* = RX + r_j d_j',$$

where  $r_j$  is the  $j^{\text{th}}$  column of  $R$  and

$$d_j = x - x_j.$$

Thus exchanging  $x_j$  for  $x$  transforms the information as

$$(X'VX)^* = X'VX - A_j B_j'$$

where

$$\begin{aligned} A_j &= \begin{bmatrix} d_j - f_j^* & d_j \\ d_j & -f_j^* - (z_j^* + 1)d_j \end{bmatrix} \\ B_j &= \begin{bmatrix} d_j - f_j^* & d_j \\ d_j & -f_j^* - (z_j^* + 1)d_j \end{bmatrix} \end{aligned} \quad (32)$$

and

$$\begin{aligned} f_j^* &= X'R'r_j = X'V_j \\ z_j^* &= r_j'r_j = V_{jj}. \end{aligned}$$

Similarly, when  $x_j$  is interchanged with  $x_k$  we have

$$(RX)^* = RX - d_{rjk} d_{xjk}'$$

where

$$\begin{aligned} d_{rjk} &= r_j - r_k \\ d_{xjk} &= x_j - x_k \end{aligned}$$

so that

$$(X'VX)^* = X'VX + A_{jk} B_{jk}'$$

where

$$\begin{aligned} A_{jk} &= \begin{bmatrix} d_{xjk} - f_{jk}^* & d_{xjk} \\ d_{xjk} & (z_{jk}^* - 1)d_{xjk} - f_{jk}^* \end{bmatrix} \\ B_{jk} &= \begin{bmatrix} d_{xjk} - f_{jk}^* & d_{xjk} \\ d_{xjk} & (z_{jk}^* - 1)d_{xjk} - f_{jk}^* \end{bmatrix} \end{aligned} \quad (33)$$

and

$$\begin{aligned} f_{jk}^* &= X'V_j - x'V_k \\ z_{jk}^* &= V_{jj} + V_{kk} - 2V_{jk}. \end{aligned}$$

In relating equations (32) and (33) back to (29) and (31), respectively, note that

$$\begin{aligned} f_j^* &= x_{1j} - f_j, \\ z_j^* &= 1 - z_j, \text{ and} \\ z_{jk}^* &= 2 - z_{jk}. \end{aligned}$$

**Initialization:** Chain together design points from  $C$  to obtain an initial design  $X$ . Initialize  $M$  to  $X'(I - Z(Z'Z)^{-1}Z)X$ .

do while ( $|M|$  improves) {

do while ( $|M|$  improves) {

**Exchange step:** For each of the design points  $\mathbf{x}_{ij}$  with the  $k$  smallest variances, exchange  $\mathbf{x}_{ij}$  with  $\mathbf{x} \in C$  for which the multiplicative change  $\Delta_E(\mathbf{x}_{ij}, \mathbf{x})$  in  $|M|$  is maximized.  
}

**Interchange step:** For each pair of design points  $\mathbf{x}_{ij}$  and  $\mathbf{x}_{kl}$ , interchange  $\mathbf{x}_{ij}$  and  $\mathbf{x}_{kl}$  if the multiplicative change  $\Delta_I(\mathbf{x}_{ij}, \mathbf{x}_{kl})$  in  $|M|$  is greater than 1.  
}

Figure 1: CN Algorithm for Optimal Block Designs

**Initialization:** Randomly select design points  $X$  from  $C$  and initialize  $M$  to  $X'(I - P)X + \alpha I$ .

do while ( $|M|$  improves) {

do while ( $|M|$  improves) {

**Exchange step:** For each of the design points  $\mathbf{x}_j$  with the  $k$  smallest variances, exchange  $\mathbf{x}_j$  with  $\mathbf{x} \in C$  for which the multiplicative change

$$\Delta_E(\mathbf{x}_j, \mathbf{x}) = |I_2 + B'_j M^{-1} A_j|$$

in  $|M|$  is maximized.  
}

**Interchange step:** For each pair of design points  $\mathbf{x}_j$  and  $\mathbf{x}_k$ , interchange  $\mathbf{x}_j$  and  $\mathbf{x}_k$  if the multiplicative change

$$\Delta_I(\mathbf{x}_j, \mathbf{x}_k) = |I_2 - B'_{jk} M^{-1} A_{jk}|$$

in  $|M|$  is greater than 1.  
}

Figure 2: COVMAX Algorithm for Optimal Designs with Covariates

$t$	$b$	$k$	$D_S$	$D_0$	$D_R$	$t$	$b$	$k$	$D_S$	$D_0$	$D_R$
3	3	2	*	*	*	7	84	3	*	*	*
4	4	3	*	*	*		91	3	9.02	*	*
	6	2	*	*	*	8	8	7	*	*	*
5	5	4	*	*	*		14	4	6.01	*	*
	10	2	*	*	*		28	2	*	*	*
		3	5.00	*	*			4	7.38	*	*
6	6	5	*	*	*			6	9.73	*	*
	10	3	5.18	*	*		42	4	8.71	*	*
	15	2	*	*	*		56	3	7.90	*	*
		4	7.39	*	*			4	8.77	*	*
	20	3	6.17	*	*			5	10.01	*	*
	30	3	7.39	*	*		70	4	9.22	*	*
	40	3	7.96	*	*	9	9	8	*	*	*
	50	3	8.41	*	*		12	3	4.38	*	*
	60	3	8.37	*	*			6	*	*	*
	70	3	9.08	*	*		18	4	6.37	*	*
	80	3	8.66	*	*			5	7.40	8.31	*
7	7	3	3.25	*	*		24	3	5.57	6.47	*
		4	5.27	*	*		36	2	*	*	*
		6	*	*	*			3	6.59	*	*
	14	3	5.27	*	*			4	7.18	*	*
	21	2	*	*	*			7	10.48	*	*
		3	6.49	*	*		48	3	7.45	*	*
		5	8.90	*	*		54	4	8.56	*	*
	28	3	6.66	*	*		60	3	7.90	*	*
	35	3	7.51	*	*		72	3	8.27	*	*
		4	8.50	*	*			4	8.80	*	*
	42	3	7.19	*	*		84	3	8.28	8.98	*
	49	3	8.19	*	*		90	4	9.12	*	*
	56	3	8.46	*	*		96	3	8.33	*	*
	63	3	7.78	*	*		108	3	8.39	*	*
	70	3	8.00	*	*		120	3	9.69	9.00	*
	77	3	8.69	*	*						

Table 1: Negative log deficiencies for block designs found by various search algorithms,  $t=3-9$

$t$	$b$	$k$	$D_S$	$D_0$	$D_R$	$t$	$b$	$k$	$D_S$	$D_0$	$D_R$		
10	10	9	*	*	*	12	22	6	7.60	8.79	9.20		
		15	4	5.88	6.80		*	33	4	6.68	*	*	
			6	7.71	*		*	44	3	6.00	7.36	*	
		18	5	6.93	8.19		*		6	8.79	9.89	10.59	
		30	3	5.90	*		*	66	4	8.07	9.56	*	
			4	7.10	8.19		*		6	9.61	11.40	*	
		36	5	8.48	9.57		*	88	3	7.36	*	*	
		45	2	*	*		*	99	4	8.68	10.38	*	
			4	7.50	*		*	13	13	4	4.41	5.53	5.53
			8	10.58	*		*			9	8.20	*	*
		54	5	9.00	*		*		26	3	4.97	6.22	*
		60	3	7.10	*		*			4	6.01	7.21	7.61
			4	8.19	*		*			6	7.50	8.19	9.04
		72	5	9.57	*		*		39	4	6.72	8.42	8.42
		75	4	8.52	*		*			5	7.84	9.04	9.04
		90	3	7.90	*		*		52	3	6.53	7.61	*
			4	8.78	*		*			4	7.40	8.31	*
	120	3	8.47	*	*		6		8.76	10.43	10.83		
11	11	5	5.58	6.40	6.40	65	4	7.84	9.45	*			
		6	6.28	6.99	7.21	78	3	7.18	8.42	*			
		10	*	*	*		4	8.02	9.12	*			
		22	5	6.98	8.08	*		5	8.89	10.43	*		
		33	5	7.59	9.30	*		6	9.70	11.24	*		
		44	5	8.27	9.87	*	91	4	8.33	9.71	*		
		55	2	*	*	*	104	3	7.49	*	*		
			3	6.67	*	*		4	8.60	9.69	*		
			4	7.91	*	*	117	4	8.92	10.22	*		
			5	8.71	*	*	130	4	8.89	10.14	*		
		66	5	9.18	*	*	14	26	7	7.96	8.65	9.13	
		77	5	9.38	*	*			52	7	9.23	10.51	11.02
		88	5	9.55	*	*			78	7	10.04	11.83	12.24
		110	3	8.20	*	*			91	4	8.17	9.63	*
			4	8.98	*	*			6	9.52	11.47	11.87	

Table 2: Negative log deficiencies for block designs found by various search algorithms,  $t=10-14$

$t$	$b$	$k$	$D_S$	$D_0$	$D_R$	t	b	k	$D_S$	$D_0$	$D_R$	
15	15	7	6.50	7.61	*	16	96	5	8.72	10.62	11.03	
		8	7.30	7.74	8.34			6	9.24	10.92	11.14	
	30	7	7.94	8.64	9.00	100	4	7.94	9.17	10.09		
		35	3	4.87	6.66			6.66	104	6	9.47	11.30
	60	6	7.64	8.64	8.64	17	34	8	8.30	8.93	9.07	
			42	5	7.43			8.53	8.53	68	4	7.17
	45	7	8.81	9.96	9.96			5	7.93	8.66	9.58	
			60	7	9.29			10.54	11.23	8	9.73	10.83
	63	5	8.24	9.15	9.85	18	34	9	8.61	9.03	9.51	
			70	3	6.45			8.05	*	51	6	7.85
	75	6	8.97	10.36	10.58			68	9	9.80	10.90	11.07
			84	5	8.53			10.83	10.83	19	19	9
	105	4	8.38	10.25	*			10	7.77	8.09	8.26	
			5	9.20	10.58			*	38	9	8.75	9.35
	16	16	6	5.89	6.38	7.00			57	3	5.32	7.39
10			8.15	9.20	*	4			6.39	7.40	8.09	
20		4	4.89	5.76	5.76			6	7.97	9.11	9.00	
			24	6	6.87			7.56	7.46	7	8.52	9.09
30		8	8.56	8.89	9.26			9	9.39	10.28	10.49	
			32	6	7.29			8.39	8.25	20	38	10
40		4	6.31	7.34	7.85			5	7.76	8.62	8.82	
			6	7.84	8.47			8.70	21	21	5	4.85
48		5	7.50	8.14	8.95			7	6.97	7.30	7.50	
			6	8.15	8.74			9.20	35	9	8.21	8.64
56		6	8.51	9.84	9.66			5	6.44	7.29	7.38	
			60	4	7.05			8.15	9.06	6	7.19	7.65
80		8	9.47	10.76	11.05			10	9.05	9.32	9.74	
			64	6	8.63			10.33	10.11	56	6	7.64
72		6	8.82	9.88	10.16			7	8.42	8.80	9.09	
	6		8.82	9.88	10.16			60	7	8.42	8.80	9.09
88	3	6.64	8.25	*			5	7.13	8.10	8.19		
		4	7.63	8.95			9.64	70	6	8.14	8.67	9.05
6	9.03	10.37	10.78				5	7.76	8.60	8.77		
		7	9.66				10.89	11.23	6	8.42	9.27	9.41
88	6	9.47	10.56	10.97			11	9.21	9.67	9.84		

Table 3: Negative log deficiencies for block designs found by various search algorithms,  $t=15-22$

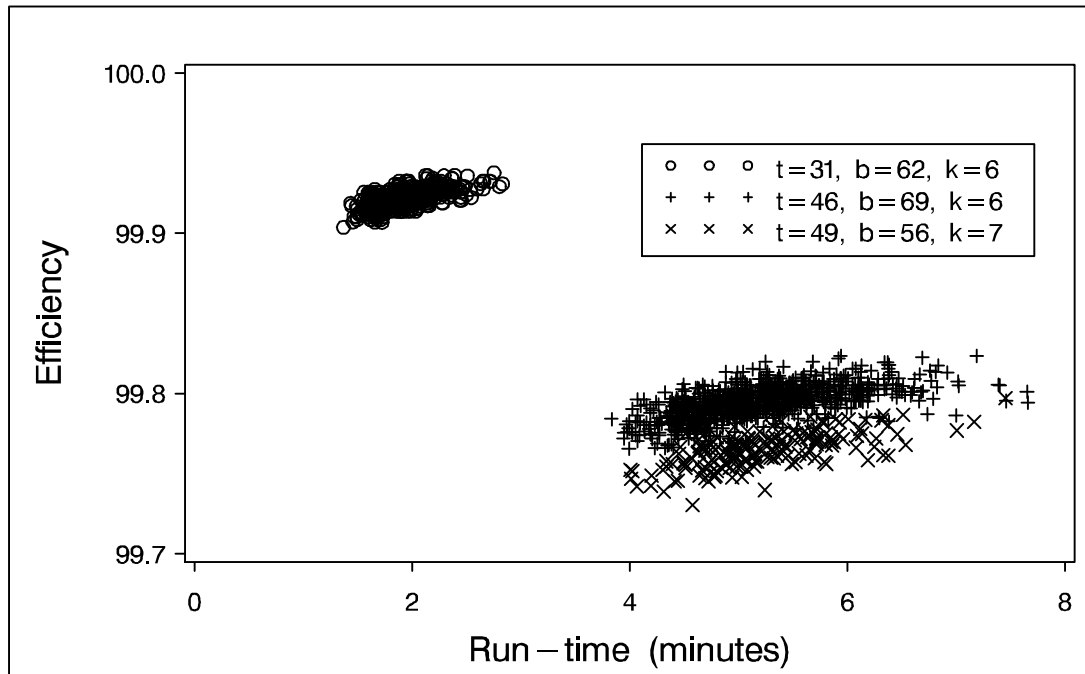


Figure 3: Efficiency versus run-time for three block designs

Number of Treatments	Covariate Model	D-Efficiency	A-Efficiency
2	Linear	100.00	100.00
	Quadratic	99.59	99.59
3	Linear	98.58	98.18
	Quadratic	97.32	96.83
4	Linear	97.29	95.98
	Quadratic	95.74	94.54
5	Linear	99.72	99.72
	Quadratic	91.66	91.13
6	Linear	94.05	89.66
	Quadratic	85.57	80.61

Table 4: Efficiencies for Harville's Designs

Treatment	Standard			
	Mean	Deviation	Skewness	Kurtosis
1	3.70	5.21	0.77	-0.94
2	3.69	5.21	0.77	-0.93
3	3.67	5.20	0.78	-0.92
4	3.66	5.20	0.78	-0.92
5	3.68	5.20	0.77	-0.93

Table 5: Within-group Moments of Covariate for Optimal Design

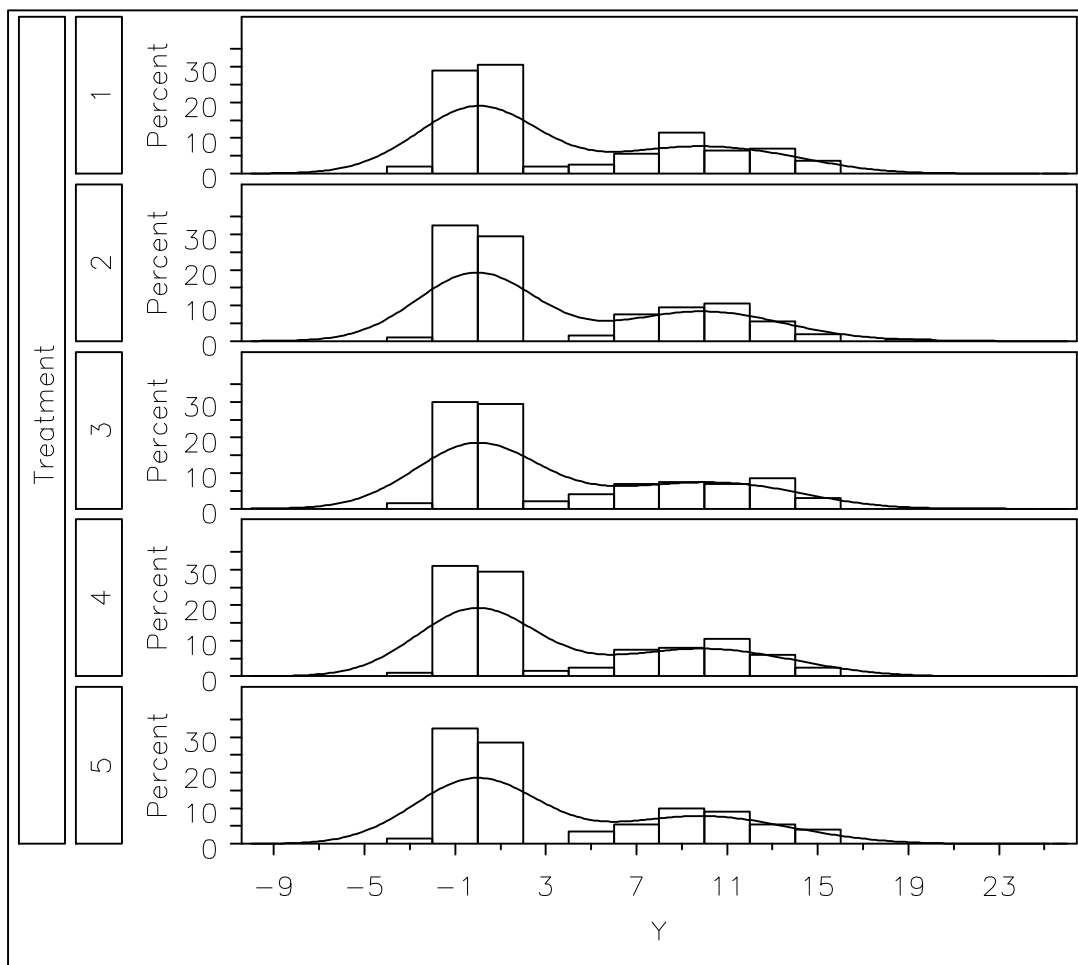


Figure 4: Within-group Distribution of Covariates for Optimal Design

3	7	2	6
5	1	6	3
1	7	5	2
4	2	3	5
7	6	5	4
2	1	4	6
7	4	3	1

Figure 5: Neighbor-balanced BIBD for  $t = b = 7$ ,  $k = 4$ , found by COVMAX