

# Optimal Compound Orthogonal Arrays and Single Arrays for Robust Parameter Design Experiments

Yu ZHU

Department of Statistics  
Purdue University  
West Lafayette, IN 47907  
(yuzhu@stat.purdue.edu)

Peng ZENG

Department of Mathematics  
and Statistics  
Auburn University  
Auburn, AL 36849  
(zengpen@auburn.edu)

Kristofer JENNINGS

Department of Statistics  
Purdue University  
West Lafayette, IN 47907  
(jennings@stat.purdue.edu)

Compound orthogonal arrays (COAs) and single arrays are alternatives to the inner–outer arrays advocated by Taguchi for robust parameter design experiments. A criterion based on the wordtype pattern and strength of COAs is proposed to select optimal COAs. Single arrays are classified into prodigal single arrays (PSAs) and economical single arrays (ESAs) according to their relative estimation capacities, and various optimality criteria, again based on the wordtype pattern, are proposed for selecting optimal single arrays. Useful optimal COAs, PSAs, and ESAs are constructed and tabulated as convenient references for experimenters in practice.

KEY WORDS: Compound orthogonal array; Robust parameter design; Single array.

## 1. INTRODUCTION

Robust parameter design (or, briefly, parameter design) was originally proposed by Taguchi (1986) as an engineering strategy for quality improvement in industrial systems. Factors that affect a system can be classified into two types: *control factors* and *noise factors*. Control factors are variables with levels that are adjustable, whereas noise factors are variables with levels that are hard or impossible to control in a system's normal operational state. In a parameter design experiment, both control and noise factors are varied systematically. The key of parameter design is to investigate the effects of control factors, noise factors, and their interactions. Then some control factor settings are chosen to simultaneously bring the system's mean response on target and reduce the performance variation caused by noise factors. (For comprehensive reviews on parameter design, see Nair 1992; Steinberg 1996.)

Taguchi originally proposed using inner–outer arrays, or *cross-arrays*, for parameter design. A cross-array is the cross-product of an orthogonal array for control factors (or, briefly, a control array) and an orthogonal array for noise factors (or, briefly, a noise array). After data are generated from an experiment using a cross-array, the response mean and variance at each setting of the control factors are calculated, and the location and dispersion effects are identified using the *location-dispersion modeling* approach (Vining and Myers 1990). Due to concerns over the run size and flexibility of cross-arrays, Lucas (1989), Welch, Yu, Kang, and Sacks (1990), and Shoemaker, Tsui, and Wu (1991) proposed using combined arrays, or *single arrays*, as an alternative to cross-arrays. A single array is an ordinary orthogonal array that accommodates both control

and noise factors and does not necessarily have the “crossing” structure of cross-arrays. In analyzing data generated from an experiment using a single array, the usual *response modeling* approach directly models the response as a function of control factors, noise factors, and their interactions. Control-by-noise plots and the transmitted variance model are then used to identify location and dispersion effects. The idea of response modeling was first hinted at by Easterling (1985).

The advantages and disadvantages of cross-array and single-array techniques have been discussed by Nair (1992) and Steinberg (1996). Recently, a fair amount of effort has been devoted to the selection of optimal experimental plans for parameter design. In a series of articles, Rosenbaum (1994, 1996, 1999) generalized cross-arrays to compound orthogonal arrays (COAs) and provided justification for using COAs in parameter design experiments. Designs using COAs do not require the rigid crossing structure of cross-arrays; as a result, different arrays can be used for noise factors at different settings of control factors. A detailed description of the properties of COAs is given in Section 3.1. Hedayat and Stufken (1999) studied the basic properties of COAs and constructed tables of COAs with fewer than six control factors and noise factors. However, the optimal selection of COAs has not yet been addressed in the literature.

Although single arrays are combinatorially the same as ordinary orthogonal arrays, their optimal selection is not straightforward, due to the presence of the two different types of factors.

Bingham and Sitter (2003) proposed a minimum-aberration type of criterion based on modified wordlengths and constructed tables of small arrays applicable in split-plot parameter design experiments. These tables show that the criterion is not sensitive in discriminating single arrays when their sizes are relatively large. Wu and Zhu (2003) developed a general framework for selecting optimal single arrays, again using a minimum-aberration type of criterion. The framework becomes too complicated when high-order effects are taken into consideration. Thus, a simple and direct approach is still needed for the selection of optimal single arrays.

In this article we address the optimal selection of COAs and single arrays. The article is organized as follows. Section 2 briefly reviews fractional factorial designs with two groups of factors. Section 3 introduces COAs and proposes a minimum-aberration criterion for selecting optimal COAs. Section 4 classifies single arrays into economical and prodigal single arrays, proposes several criteria for selecting optimal single arrays, and tabulates and discusses optimal economical and prodigal single arrays. Section 5 gives some concluding remarks. In this article we focus on COAs and single arrays that are regular two-level fractional factorial designs.

## 2. $2^{(l_1 + l_2) - m}$ DESIGNS WITH TWO GROUPS OF FACTORS

Regular two-level fractional factorial designs (i.e.,  $2^{l-m}$  designs) are generated by defining relations (or defining words) among experimental factors. The collection of all possible defining words of a design  $d$  is referred to as its *defining contrast subgroup*, denoted by  $\mathcal{G}$ . Let  $W_i$  be the number of defining words of length  $i$  in  $\mathcal{G}$  for  $1 \leq i \leq l$  and  $W = (W_0, W_1, W_2, \dots, W_l)$ . Then  $W$  is called the *wordlength pattern* of  $d$ . The *resolution* of  $d$  is the smallest positive integer  $i$  such that  $W_i > 0$ . If  $d$  has resolution  $R$ , then it is well known that the strength of  $d$  is  $R - 1$ . (See Rao 1947 for the definition of “strength” for general orthogonal arrays.) Suppose that  $d_1$  and  $d_2$  are two designs. Design  $d_1$  is said to have less aberration than design  $d_2$  if  $W_{i_0}(d_1) < W_{i_0}(d_2)$ , where  $i_0$  is the smallest integer  $i$  such that  $W_i(d_1) \neq W_i(d_2)$ . If there does not exist a design with less aberration than  $d_1$ , then  $d_1$  is said to have *minimum aberration* (MA) (Fries and Hunter 1980). A main effect or a two-factor interaction (2fi) is said to be *clear*, or clearly estimable, if it is not aliased with any other main effects or 2fi’s, and *eligible* if it is not clear but only aliased with some other 2fi’s (Wu and Chen 1992).

For  $2^{l-m}$  designs, experimental factors are treated without further distinction. But this is not proper when multiple groups or types of factors are involved, for example, in parameter design experiments where factors are divided into control factors and noise factors. In this article we consider only the case with two groups of factors, denoted as group I and group II. Let  $l_1$  and  $l_2$  be the number of factors in group I and group II. A  $2_\tau^{(l_1 + l_2) - p}$  design  $d$  is a two-level fractional factorial design with  $l_1$  columns assigned to group I factors and the remaining  $l_2$  columns assigned to group II factors. The subscript  $\tau$  represents the assignment of columns and is usually suppressed. The design  $d$  is also determined by its defining contrast subgroup  $\mathcal{G}$ . However, the wordlength pattern  $W$  does not reflect the fact

that defining words with the same length can consist of different numbers of group I and group II factors. Zhu (2003) proposed using the *wordtype pattern* matrix,  $(A_{i,j})_{0 \leq i \leq l_1, 0 \leq j \leq l_2}$ , to characterize the aliasing pattern of  $\mathcal{G}$ , where  $A_{i,j}$  is the number of defining words in  $\mathcal{G}$  involving  $i$  group I factors and  $j$  group II factors. In a parameter design experiment involving  $l_c$  control factors and  $l_n$  noise factors, we assume that group I consists of the control factors and group II consists of the noise factors. If we consider only two-level regular fractional factorial designs as possible experimental plans, then cross-arrays, COAs, and single arrays are in fact  $2^{(l_c + l_n) - m}$  designs with or without additional constraints. As a convention in this article we use capital letters (e.g.,  $A, B$ ) to represent control factors and lower-case letters (e.g.,  $a, b$ ) to represent noise factors.

## 3. OPTIMAL COMPOUND ORTHOGONAL ARRAYS

### 3.1 Compound Orthogonal Arrays and Maximum Strength

Let  $OA(N, l, 2, t)$  denote a two-level orthogonal array with  $N$  rows,  $l$  columns, and strength  $t$  (Rao 1947). A COA with parameters  $N_c, N_n, l_c, l_n, t_c, t_n$ , and  $t_a$  is an  $N_c N_n \times (l_c + l_n)$  orthogonal array with the following properties:

- P1. The first  $l_c$  columns are assigned to control factors and consist of  $N_n$  replications of an  $OA(N_c, l_c, 2, t_c)$ .
- P2. The remaining  $l_n$  columns are assigned to noise factors, and for each setting of the  $l_c$  control factors, the corresponding settings of the  $l_n$  noise factors form an  $OA_{c_i}(N_n, l_n, 2, t_n)$ , where the subscript  $c_i$  for  $1 \leq i \leq N_c$  indexes the particular control factor setting.
- P3. The entire array has strength  $t_a$ .

Let  $T = (t_c, t_n, t_a)$ . We call  $T$  the *strength vector* of the COA. It can be verified that  $\min(t_c, t_n) \leq t_a \leq (t_c + t_n)$  (Hedayat and Stufken 1999). When all the  $OA_{c_i}(N_n, l_n, 2, t_n)$ ’s are identical for  $1 \leq i \leq N_c$ , the COA is a cross-array. A COA is said to be *regular* if it is a regular  $2^{(l_c + l_n) - m}$  design that satisfies P1–P3. As mentioned previously, we consider only regular COAs in this article.

*Example 1.* Suppose that a 32-run experiment involves four control factors ( $A, B, C, D$ ) and three noise factors ( $a, b, c$ ). Consider the following two  $2^{(4+3)-2}$  designs:  $d_1: I = ABCD = ABabc = CDabc$  and  $d_2: I = ABCD = abc = ABCDabc$ . Both  $d_1$  and  $d_2$  have 32 rows and 7 columns, with the first 4 assigned to the control factors and the remaining 3 assigned to the noise factors. In  $d_1$ , the control factor columns consist of four copies of an eight-run  $2^{4-1}$  resolution IV design defined by  $I = ABCD$ . At each fixed setting of  $A, B, C$  and  $D$ , the corresponding settings of the noise factors form a four-run resolution III design defined by  $AB = CD = abc$ . For example, at the setting  $(1, -1, 1, -1)$  of the control factors, the corresponding settings of  $a, b$ , and  $c$  are generated by  $-I = abc$ , which include  $(-1, -1, -1), (-1, 1, 1), (1, -1, 1)$ , and  $(1, 1, -1)$ . So  $d_1$  is a COA with  $N_c = 8, N_n = 4, l_c = 4, l_n = 3, t_c = 3, t_n = 2$ , and  $t_a = 3$ , and  $T(d_1) = (3, 2, 3)$ . Similarly, it can be verified that  $d_2$  is a COA with  $T(d_2) = (3, 2, 2)$ . Note that in  $d_2$ , the noise arrays, which are generated by  $I = abc$ , are identical at different settings of the control factors. Thus  $d_2$  is a cross-array.

However,  $d_1$  is not a cross-array, but has higher overall strength than  $d_2$ . The wordtype patterns of  $d_1$  are  $A_{0,0} = 1$ ,  $A_{2,3} = 2$ ,  $A_{4,0} = 1$ , and  $A_{i,j} = 0$  otherwise; the wordtype patterns of  $d_2$  are  $A_{0,0} = 1$ ,  $A_{0,3} = A_{4,0} = A_{4,3} = 1$ , and  $A_{i,j} = 0$  otherwise.

Let  $d$  be a  $2^{(l_c+l_n)-m}$  COA with wordtype pattern matrix  $(A_{i,j})$ . Considering the relationship between strength and resolution, we can verify that

$$t_c = \min(l_c, \min\{i - 1 : A_{i,0} \neq 0 \text{ and } i \geq 1\}),$$

$$t_n = \min(l_n, \min\{j - 1 : A_{i,j} \neq 0 \text{ and } j \geq 1\}),$$

and

$$t_a = \min\{i + j - 1 : A_{i,j} \neq 0 \text{ and } i + j \geq 1\}.$$

Hence  $(A_{i,j})$  determines  $T(d)$ . In the foregoing definition of COA, there is no specific restriction on  $t_n$ , so  $t_n$  can take on any nonnegative integer value. Rosenbaum (1994, 1996) proposed COAs as a generalization of cross-arrays and pointed out that COAs with  $t_n < 2$  are not able to identify dispersion effects and are unlikely to be useful (see thm. 1 and sec. 3.1 of Rosenbaum 1996). In other words, COAs with  $t_n \leq 1$  are not a reasonable generalization of cross-arrays. When investigating the maximum strength of COAs from a theoretical perspective, Hedayat and Stufken (1999) included COAs with  $t_n \leq 1$ .

In this article we are more interested in COAs that are useful in parameter design; therefore, in addition to the three properties given earlier, we further require that COAs satisfy the following condition:

$$P4. \quad t_c \geq \min(l_c, 2) \text{ and } t_n \geq \min(l_n, 2).$$

For regular  $2^{(l_c+l_n)-m}$  COAs with resolution III or higher ( $t_a \geq 2$ ), the condition  $t_c \geq \min(l_c, 2)$  is automatically satisfied. When  $l_n > 2$ , the condition  $t_n \geq \min(l_n, 2)$  requires that the noise arrays of a COA have strength at least 2. In terms of wordtype patterns, the condition  $t_n \geq \min(l_n, 2)$  is equivalent to  $A_{i,1} = A_{i,2} = 0$  for  $1 \leq i \leq l_c$ . Clearly, cross-arrays are COAs. Both  $d_1$  and  $d_2$  in Example 1 satisfy P4, so they are still COAs.

For given  $l_c$ ,  $l_n$ , and run size  $2^k$ , COAs do not always exist. Define

$$\mathcal{S}(2^k) = \{(i, j) : \lceil \log_2(i + 1) \rceil + \lceil \log_2(j + 1) \rceil \leq k\}, \quad (1)$$

where  $\lceil x \rceil$  represents the smallest integer greater than or equal to  $x$ .

*Proposition 1.* The necessary and sufficient condition for the existence of a COA with  $l_c$  control factors,  $l_n$  noise factors, and  $2^k$  runs is  $(l_c, l_n) \in \mathcal{S}(2^k)$ .

The proof of Proposition 1 is given in the Appendix. When  $k = 4$ ,  $\mathcal{S}(2^4) = \{(1, j) : 1 \leq j \leq 7\} \cup \{(i, 1) : 1 \leq i \leq 7\} \cup \{(2, 2), (2, 3), (3, 2), (3, 3)\}$ . Because  $(3, 4)$  is not in  $\mathcal{S}(2^4)$ , there does not exist a 16-run COA with 3 control factors and 4 noise factors. Note that the necessary and sufficient condition for the existence of cross-arrays is exactly the same as that for COAs.

Suppose that  $d$  is a COA with  $T(d) = (t_c(d), t_n(d), t_a(d))$ . The array  $d$  is said to have *maximum strength* if there does not exist another COA  $\tilde{d}$  with the same run size,  $l_c$ , and  $l_n$  as  $d$  such that  $T(\tilde{d}) \geq T(d)$  componentwise and at least one of the inequalities is strict. The preference for COAs with maximum strength is justified by theorem 1 of Rosenbaum (1996) and an

observation of Hedayat and Stufken (1999) that has further improved part c of theorem 1 of Rosenbaum (1996) (see the beginning of sec. 2 of Hedayat and Stufken 1999). When the response modeling approach is used in data analysis, COAs with maximum strength are preferred, according to section 4 of Rosenbaum (1996). In Example 1,  $T(d_1) = (3, 2, 3)$  and  $T(d_2) = (3, 2, 2)$ . Clearly,  $T(d_1) \geq T(d_2)$  and  $t_a(d_1) > t_a(d_2)$ . Thus design  $d_2$ , a cross-array, does not achieve maximum strength. In fact, we can show that  $d_1$  attains maximum strength as a COA with 32 runs, 4 control factors, and 3 noise factors.

### 3.2 $W_c$ -Aberration and Optimal Compound Orthogonal Arrays

The strength vector  $T$  of a COA can be considered an extension of the resolution  $R$  of a fractional factorial design. Different COAs may share the same strength vector and must be further discriminated.

*Example 2.* Consider a 64-run parameter design experiment involving 4 control factors ( $A, B, C, D$ ) and 6 noise factors ( $a, b, c, d, e, f$ ). Four possible  $2^{(4+6)-4}$  COAs are given here. Due to space limitations, only one set of independent defining words is listed for each array:

$$d_3 : ABCD, Dabd, Dace, Dbcf;$$

$$d_4 : ABCD, abde, ABacd, ACabf;$$

$$d_5 : ABCD, abce, abdf, ACacd;$$

$$d_6 : ABCD, abd, ace, bcf.$$

Array  $d_3$  is the design given in table 1 of Rosenbaum (1996), array  $d_4$  is a MA  $2^{10-4}$  design with the distinction between control factors and noise factors neglected, and array  $d_6$  is a cross-array. The strength vectors of the four COAs are  $T(d_3) = T(d_4) = T(d_5) = (3, 2, 3)$  and  $T(d_6) = (3, 2, 2)$ . It can be shown that  $d_3$ ,  $d_4$ , and  $d_5$  all have achieved the maximum strength for COAs with 64 rows, 4 control factors, and 6 noise factors; however, they are still quite different from one another, as we show later, so further discrimination among them is needed to select the best plan for the experiment.

The defining contrast subgroups and wordtype pattern matrices of the foregoing arrays can be derived from independent defining words. It can be verified that the wordtype pattern matrices differ. In what follows, we further compare  $d_3$ ,  $d_4$ , and  $d_5$  in terms of their detailed aliasing patterns. For simplicity, we assume that effects of order 3 or higher are negligible. Thus we need to consider only the defining words with length  $< 5$ , which are  $\{ABCD, Dabd, Dace, Dbcf, Ddef, bcde, acdf, abef\}$  for  $d_3$ ,  $\{ABCD, abde\}$  for  $d_4$ , and  $\{ABCD, abce, abdf, cdef\}$  for  $d_5$ . Because all the three arrays have resolution IV, their control and noise main effects are clear. Because  $ABCD$  appears in  $d_3$ ,  $d_4$ , and  $d_5$ , the control-by-control 2fi's in all the three arrays are eligible. Consider the 24 possible control-by-noise 2fi's. In  $d_3$ , the 18 control-by-noise 2fi's between  $\{A, B, C\}$  and  $\{a, b, c, d, e, f\}$  are clear, and the 6 control-by-noise 2fi's between  $\{D\}$  and  $\{a, b, c, d, e, f\}$  are aliased with some noise-by-noise 2fi's and thus are only eligible. In  $d_4$ , all of the 24 control-by-noise 2fi's are clear; in addition, the noise-by-noise 2fi's  $ac$ ,  $bc$ ,  $cd$ ,  $ce$ ,  $cf$ ,  $af$ ,  $bf$ ,  $df$ , and  $ef$  are also clear; and the other 6

noise-by-noise 2fi's are eligible. In  $d_5$ , all 24 control-by-noise 2fi's are clear, but all of the noise-by-noise 2fi's are only eligible. Heuristically,  $d_4$  is the best among the three arrays according to their aliasing patterns, followed by  $d_5$  and  $d_3$ . Thus,  $d_4$  should be selected for the experiment.

Example 2 shows that COAs with maximum strength may not be unique and can be further discriminated by their wordtype patterns. Let

$$\overbrace{C \dots C}^i \overbrace{n \dots n}^j$$

represent a defining word consisting of  $i$  control factors and  $j$  noise factors, which is said to be of type  $(i, j)$ . Note that  $A_{i,j}$  is the number of defining words of type  $(i, j)$  in the defining contrast subgroup of a COA. Defining words of length  $k$  have  $(k + 1)$  different types:  $(k, 0), (k - 1, 1), \dots, (1, k - 1)$ , and  $(0, k)$ . In what follows, we rearrange  $A_{i,j}$  into a sequence that can be used to rank-order different COAs. First, we consider the defining words of length 3:  $CCC, CCn, Cnn$ , and  $nnn$ . In COAs,  $CCn$  and  $Cnn$  are not present. Because the purpose of robust parameter design encompasses both mean response optimization and variance reduction,  $CCC$  is considered more severe than  $nnn$  in terms of aliasing severity. We denote this relationship as  $CCC \triangleleft nnn$ , or  $A_{3,0} \triangleleft A_{0,3}$ .

Next we consider the defining words of length 4:  $CCCC, CCCn, CCnn, Cnnn$ , and  $nnnn$ . The defining words  $CCCn$  and  $CCnn$  are not present in COAs, and in a similar way, we regard  $CCCC$  as more severe than  $nnnn$ , that is,  $CCCC \triangleleft nnnn$ . It is reasonable that  $Cnnn$  is considered more severe than  $nnnn$ ; however, the comparison between  $CCCC$  and  $Cnnn$  is not conclusive at first glance.  $CCCC$  causes aliasing between control-by-control 2fi's, which may hinder the identification of second-order location effects for mean response optimization, whereas  $Cnnn$  causes aliasing between control-by-noise 2fi's and noise-by-noise 2fi's, which may hinder the identification of first-order dispersion effects for variation reduction. Therefore, we regard  $Cnnn$  as more severe than  $CCCC$  and write  $Cnnn \triangleleft CCCC \triangleleft nnnn$ , or  $A_{1,3} \triangleleft A_{4,0}, \triangleleft A_{0,4}$ .

For the defining words of order 5, we have  $CCnnn \triangleleft Cnnnn \triangleleft CCCCC \triangleleft nnnnn$ , or  $A_{2,3} \triangleleft A_{1,4} \triangleleft A_{5,0} \triangleleft A_{0,5}$ . In general,  $A_{i_1, j_1} \triangleleft A_{i_2, j_2}$  if

- (a)  $i_1 + j_1 < i_2 + j_2$ ,
- (b)  $|i_1 - j_1| < |i_2 - j_2|$  and  $i_1 + j_1 = i_2 + j_2$ , or
- (c)  $i_1 > i_2$  and  $|i_1 - j_1| = |i_2 - j_2|$  and  $i_1 + j_1 = i_2 + j_2$ .

Thus, following  $\triangleleft$ , all wordtype patterns  $A_{i,j}$  can be arranged into a sequence, denoted by  $W_c$ , from the most severe to the least severe, as follows:

$$W_c = (A_{3,0}, A_{0,3}, A_{1,3}, A_{4,0}, A_{0,4}, A_{2,3}, A_{1,4}, A_{5,0}, A_{0,5}, A_{3,3}, A_{2,4}, A_{1,5}, A_{6,0}, A_{0,6}, \dots). \quad (3)$$

We refer to  $W_c$  as the wordtype pattern sequence for COAs. Note that  $A_{i,j}$  with  $0 \leq i + j \leq 2$  are constants for designs with resolution at least III, so they are not included in  $W_c$ .

Suppose that  $d_1$  and  $d_2$  are two COAs with wordtype pattern sequences  $W_c(d_1)$  and  $W_c(d_2)$ . Let  $A_{i_0, j_0}$  be the first component of  $W_c$  such that  $A_{i_0, j_0}(d_1) \neq A_{i_0, j_0}(d_2)$ . If  $A_{i_0, j_0}(d_1) < A_{i_0, j_0}(d_2)$ ,

then  $d_1$  is said to have less  $W_c$  aberration than  $d_2$ . If there does not exist a COA with less  $W_c$  aberration than  $d_1$ , then  $d_1$  is said to have minimum  $W_c$  aberration. Furthermore, if  $d_1$  has minimum  $W_c$  aberration and maximum strength, then  $d_1$  is said to be an optimal COA. The reason for including the requirement of maximum strength in the optimality criterion for COAs is that it is not necessarily true that  $T(d_1) \geq T(d_2)$  componentwise when  $d_1$  has less  $W_c$  aberration than  $d_2$ . Although all known minimum  $W_c$  aberration COAs achieve maximum strength, we are not able to prove that it is true in general. So we conjecture that minimum  $W_c$  aberration COAs also achieve maximum strength.

Example 2 (Continued). We have that

$$W_c(d_3) = (0, 0, 4, 1, 3, 0, 0, 0, 0, 4, 0, \dots),$$

$$W_c(d_4) = (0, 0, 0, 1, 1, 8, 0, 0, 0, 0, 4, \dots),$$

$$W_c(d_5) = (0, 0, 0, 1, 3, 8, 0, 0, 0, 0, 0, \dots),$$

and

$$W_c(d_6) = (0, 4, 0, 1, 3, 0, 0, 0, 0, 0, 0, \dots).$$

Sorted by increasing  $W_c$  aberration, the order of arrays becomes  $d_4, d_5, d_3$ , and  $d_6$ . A comprehensive search concludes that  $d_4$  has minimum  $W_c$  aberration and maximum strength among all 64-run COAs with 4 control factors and 6 noise factors. Thus  $d_4$  is the optimal COA and should be recommended for the experiment, followed by  $d_5$ .

### 3.3 Tables of Optimal Compound Orthogonal Arrays With $l_n \geq 3$

Because optimal COAs of 16, 32, and 64 runs are useful for parameter design experiments, it is valuable to tabulate them as a convenient reference for experimenters in practice. The optimal COAs with  $2^k$  runs and one or two noise factors can be easily constructed as follows. Because  $l_n \leq 2$ ,  $W_c$  is reduced to  $(A_{i,0})_{3 \leq i \leq l_c}$  after the zero components are removed. For  $(l_c, 1)$  or  $(l_c, 2)$  belonging to  $S(2^k)$ , the optimal COA is a cross-array, where the control array is either a  $2^{l_c}$  full factorial design if  $1 \leq l_c \leq k - l_n$  or a  $2^{l_c - (l_c + l_n - k)}$  MA design if  $l_c > k - l_n$ , and the noise array is a  $2^{l_n}$  full factorial design.

When  $l_n \geq 3$ , how to construct optimal COAs with  $(l_c, l_n) \in S(2^k)$  is not clear. We have conducted an exhaustive search with complete isomorphism checking to select optimal COAs. Two  $2^{(l_1 + l_2) - p}$  designs  $d$  and  $d'$  are said to be isomorphic if there exists a relabeling of the factors of  $d$  that transforms  $\mathcal{G}(d)$  to  $\mathcal{G}(d')$ , where  $\mathcal{G}(d)$  and  $\mathcal{G}(d')$  are the defining contrast subgroups of  $d$  and  $d'$ .

Our search procedure comprises three steps: (1) constructing a list of all nonisomorphic  $2^{(l_1 + l_2) - p}$  designs; (2) sorting all nonisomorphic designs according to the optimality criterion for COAs for all values of  $l_c, l_n$ , and  $k$  ( $k = 4, 5, 6$ ); and (3) choosing the optimal COAs. For 16-run and 32-run designs, we have used the lists of nonisomorphic  $2^{l-p}$  designs generated by Chen, Sun, and Xu (1993), where factors are not separated into two groups. For 64-run designs, we have constructed a list of all nonisomorphic  $2^{(l_1 + l_2) - p}$  designs from scratch for  $l_1 + l_2$  up to 16.

The final output of our search is reported in Tables 1 and 2. Table 1 contains all 16-run and 32-run optimal COAs with

Table 1. The 16-run and 32-run optimal COAs

| Array                | Generators  | Strength  | Clear effects    |
|----------------------|---|-----------|------------------|
| <b>16-run</b>        |   |           |                  |
| (1, 4)*              | <i>Aabcd</i>  | (1, 3, 4) | (1, 4, 0, 4, 6)  |
| (2, 3)*              | <i>ABabc</i>  | (2, 2, 4) | (2, 3, 1, 6, 3)  |
| (1, 5) <sup>o</sup>  | <i>Aabd Aace</i>  | (1, 2, 3) | (1, 5, 0, 0, 0)  |
| (3, 3) <sup>o</sup>  | <i>ABC Aabc</i>   | (2, 2, 2) | (0, 3, 0, 6, 0)  |
| (1, 6) <sup>o</sup>  | <i>Aabd Aace Abcf</i>   | (1, 2, 3) | (1, 6, 0, 0, 0)  |
| (1, 7) <sup>o</sup>  | <i>Aabd Aace Abcf abcg</i>                                      | (1, 2, 3) | (1, 7, 0, 0, 0)  |
| <b>32-run</b>        |   |           |                  |
| (1, 5)*              | <i>Aabcde</i>   | (1, 4, 5) | (1, 5, 0, 5, 10) |
| (2, 4)*              | <i>ABabcd</i>   | (2, 3, 5) | (2, 4, 1, 8, 6)  |
| (3, 3)*              | <i>ABCabc</i>   | (3, 2, 5) | (3, 3, 3, 9, 3)  |
| (1, 6)*              | <i>abce Aabdf</i>   | (1, 3, 3) | (1, 6, 0, 6, 9)  |
| (2, 5)*              | <i>abcd ABabe</i>   | (2, 2, 3) | (2, 5, 1, 10, 4) |
| (3, 4)*              | <i>ABC Aabcd</i>  | (2, 3, 2) | (0, 4, 0, 12, 6) |
| (4, 3)*              | <i>ABCD ABabc</i>   | (3, 2, 3) | (4, 3, 0, 12, 3) |
| (1, 7)*              | <i>abce abdf Aacd</i>   | (1, 3, 3) | (1, 7, 0, 7, 6)  |
| (2, 6)*              | <i>abce abdf ABacd</i>  | (2, 2, 3) | (2, 6, 1, 12, 0) |
| (3, 5) <sup>o</sup>  | <i>ABC Aabd Bace</i>  | (2, 2, 2) | (0, 5, 0, 9, 4)  |
| (5, 3)*              | <i>ABD ACE BCabc</i>  | (2, 2, 2) | (0, 3, 0, 15, 3) |
| (1, 8)*              | <i>abcf abdg abeh Aacde</i>                                     | (1, 3, 3) | (1, 8, 0, 8, 0)  |
| (2, 7)*              | <i>abce abdf acdg ABbcd</i>                                     | (2, 2, 3) | (2, 7, 1, 14, 0) |
| (3, 6) <sup>o</sup>  | <i>ABC Aabd Aace Bbcf</i>                                       | (2, 2, 2) | (0, 6, 0, 8, 1)  |
| (6, 3)*              | <i>ABD ACE BCF ABCabc</i>                                       | (2, 2, 2) | (0, 3, 0, 18, 3) |
| (1, 9) <sup>o</sup>  | <i>Aabe Aacf Aadg Abcdh abcdi</i>                               | (1, 2, 3) | (1, 9, 0, 0, 0)  |
| (3, 7) <sup>o</sup>  | <i>ABC Aabd Aace Abcf Babcg</i>                                 | (2, 2, 2) | (0, 7, 0, 8, 0)  |
| (7, 3) <sup>o</sup>  | <i>ABD ACE BCF ABCG Aabc</i>                                    | (2, 2, 2) | (0, 3, 0, 18, 0) |
| (1, 10) <sup>o</sup> | <i>abce abdf acdg bcdh Aabi Aacj</i>                            | (1, 2, 3) | (1, 10, 0, 0, 0) |
| (1, 11) <sup>o</sup> | <i>abce abdf acdg bcdh Aabi Aacj Aadk</i>                       | (1, 2, 3) | (1, 11, 0, 0, 0) |
| (1, 12) <sup>o</sup> | <i>abcf abdg acdh bcdi abej acek bcel Aade</i>                  | (1, 2, 3) | (1, 12, 0, 0, 0) |
| (1, 13) <sup>o</sup> | <i>Aabe Aacf Abcg abch Aadi Abdj abdk Acdl acdm</i>             | (1, 2, 3) | (1, 13, 0, 0, 0) |
| (1, 14) <sup>o</sup> | <i>Aabe Aacf Abcg abch Aadi Abdj abdk Acdl acdm bcdn</i>        | (1, 2, 3) | (1, 14, 0, 0, 0) |
| (1, 15) <sup>o</sup> | <i>Aabe Aacf Abcg abch Aadi Abdj abdk Acdl acdm bcdn Aabcdo</i> | (1, 2, 3) | (1, 15, 0, 0, 0) |

$l_n \geq 3$ , and Table 2 contains all 64-run optimal COAs with  $l_c + l_n \leq 16$  and  $l_n \geq 3$ . In both tables, every row gives an optimal COA with the columns, from left to right, giving the vector  $(l_c, l_n)$ , independent defining words (or generators), the strength vector  $T = (t_c, t_n, t_a)$ , as well as a five-tuple containing the numbers of clear control main effects, noise main effects, control-by-control 2fi's, control-by-noise 2fi's, and noise-by-noise 2fi's. For example, the optimal 32-run COA with  $l_c = 2$  and  $l_n = 7$ , denoted by  $d_1$ , is

$$(2, 7)^* \quad abce \ abdf \ acdg \ ABbcd \quad (2, 2, 3) \quad (2, 7, 1, 14, 0);$$

the optimal 32-run COA with  $l_c = 3$  and  $l_n = 6$ , denoted by  $d_2$ , is

$$(3, 6)^o \quad ABC \ Aabd \ Aace \ Bbcf \quad (2, 2, 2) \quad (0, 6, 0, 8, 1).$$

Array  $d_1$  is a resolution IV design with control array and noise arrays of strength 2. In  $d_1$ , all of the control main effects, noise main effects, control-by-control 2fi's, and control-by-noise 2fi's are clear. The 32-run cross-array with  $l_c = 2$  and  $l_n = 7$ , denoted by  $d'_1$ , is given by crossing a  $2^2$  control array and a  $2^{7-4}$  noise array. Note that the noise array in  $d'_1$  is a saturated eight-run fractional factorial design. The resolution of  $d'_1$  is III, less than that of  $d_1$ . In  $d'_1$ , the control main effects and control-by-noise 2fi's are also clear, but the noise main effects are only eligible. Therefore, in terms of both strength and the number of clear

effects,  $d_1$  is better than  $d'_1$ , as indicated by the  $\star$  on  $(2, 7)^*$ . The same interpretation applies to other arrays marked with a  $\star$  in Tables 1 and 2.

The optimal COA  $d_2$  is a resolution III design with control array and noise arrays of strength 2. In  $d_2$ , six noise main effects, eight control-by-noise 2fi's, and one noise-by-noise 2fi are clear, and control main effects are aliased with control-by-control 2fi's as well as the other control-by-noise 2fi's. Noise-by-noise 2fi's are only eligible in  $d_2$ . The corresponding 32-run cross-array, denoted by  $d'_2$ , is obtained by crossing a 4-run  $2^{3-1}$  control array and an 8-run  $2^{6-3}$  design. Arrays  $d_2$  and  $d'_2$  have the same strength vector. In  $d'_2$ , all of the control-by-noise 2fi's are clear; control main effects are aliased with some control-by-control 2fi's, noise main effects are aliased with some noise-by-noise 2fi's; and the remaining 2fi's are eligible. There clearly exists a trade-off between  $d_2$  and  $d'_2$ ; that is,  $d_2$  guarantees clear estimation of the noise main effects, whereas  $d'_2$  ensures clear estimation of all of the control-by-noise 2fi's. Due to this trade-off, the advantage of  $d'_2$  over  $d_2$  is not as clear as that of  $d_1$  over  $d'_1$ , as discussed in the preceding paragraph. We indicate this trade-off by the  $o$  on  $(3, 6)^o$ . The same interpretation applies to other arrays marked with a  $o$  in Tables 1 and 2. The trade-off also can be regarded as an indication that the array (for the given  $l_c$  and  $l_n$ ) is already too

Table 2. The 64-run optimal COAs

| Array    | Generators   | Strength  | Clear effects      |
|----------|--|-----------|--------------------|
| (1, 6)*  | <i>Aabcdef</i>   | (1, 5, 6) | (1, 6, 0, 6, 15)   |
| (2, 5)*  | <i>ABabcde</i>   | (2, 4, 6) | (2, 5, 1, 10, 10)  |
| (3, 4)*  | <i>ABCabcd</i>   | (3, 3, 6) | (3, 4, 3, 12, 6)   |
| (4, 3)*  | <i>ABCDabc</i>   | (4, 2, 6) | (4, 3, 6, 12, 3)   |
| (1, 7)*  | <i>Aabcf abdeg</i>   | (1, 3, 4) | (1, 7, 0, 7, 21)   |
| (2, 6)*  | <i>Aabce Babdf</i>   | (2, 3, 4) | (2, 6, 1, 12, 15)  |
| (3, 5)*  | <i>ABabd Cabce</i>   | (3, 2, 4) | (3, 5, 3, 15, 10)  |
| (4, 4)*  | <i>ABCD ABabcd</i>   | (3, 3, 3) | (4, 4, 0, 16, 6)   |
| (5, 3)*  | <i>ABabc ABCDE</i>   | (4, 2, 4) | (5, 3, 10, 15, 3)  |
| (1, 8)*  | <i>abcf Aabdg acdeh</i>  | (1, 3, 3) | (1, 8, 0, 8, 22)   |
| (2, 7)*  | <i>abce Aabdf Bacdg</i>  | (2, 3, 3) | (2, 7, 1, 14, 15)  |
| (3, 6)*  | <i>abce ACabd Bacdf</i>  | (3, 2, 3) | (3, 6, 3, 18, 9)   |
| (4, 5)*  | <i>ABCD ABabd ACace</i>  | (3, 2, 3) | (4, 5, 0, 20, 10)  |
| (5, 4)*  | <i>ABD ACE BCabcd</i>  | (2, 3, 2) | (0, 4, 0, 20, 6)   |
| (6, 3)*  | <i>ABCE ABDF ACDabc</i>  | (3, 2, 3) | (6, 3, 0, 18, 3)   |
| (1, 9)*  | <i>abcf Aabdg Aabeh acdei</i>                                    | (1, 3, 3) | (1, 9, 0, 9, 24)   |
| (2, 8)*  | <i>abcf Aabdg Aabeh Bacde</i>                                    | (2, 3, 3) | (2, 8, 1, 16, 16)  |
| (3, 7)*  | <i>abcf abdg Bacde ACabe</i>                                     | (3, 2, 3) | (3, 7, 3, 21, 6)   |
| (4, 6)*  | <i>ABCD ABabd ABace ACbcf</i>                                    | (3, 2, 3) | (4, 6, 0, 24, 9)   |
| (5, 5)*  | <i>ABD ACE BCabd ABCace</i>                                      | (2, 2, 2) | (0, 5, 0, 25, 10)  |
| (6, 4)*  | <i>ABD ACE BCF ABCabcd</i>                                       | (2, 3, 2) | (0, 4, 0, 24, 6)   |
| (7, 3)*  | <i>ABCE ABDF ACDG BCDabc</i>                                     | (3, 2, 3) | (7, 3, 0, 21, 3)   |
| (1, 10)* | <i>abcg abdh acei acdfj Aabef</i>                                | (1, 3, 3) | (1, 10, 0, 10, 24) |
| (2, 9)*  | <i>abcf abdg abeh acdei ABbcde</i>                               | (2, 2, 3) | (2, 9, 1, 18, 8)   |
| (3, 8)*  | <i>ABC abce Aabdf Bacdg ABbcdh</i>                               | (2, 3, 2) | (0, 8, 0, 24, 16)  |
| (4, 7)*  | <i>abce abdf ACacd BDacd ABabg</i>                               | (3, 2, 3) | (4, 7, 0, 28, 6)   |
| (5, 6)*  | <i>ABD ACE BCabd BCace ABCbcf</i>                                | (2, 2, 2) | (0, 6, 0, 30, 9)   |
| (6, 5)*  | <i>ABD ACE BCF ABCabd ABCace</i>                                 | (2, 2, 2) | (0, 5, 0, 30, 4)   |
| (7, 4)*  | <i>ABD ACE BCF ABCG Aabcd</i>                                    | (2, 3, 2) | (0, 4, 0, 28, 6)   |
| (8, 3)*  | <i>ABCE ABDF ACDG BCDH ABabc</i>                                 | (3, 2, 3) | (8, 3, 0, 24, 3)   |
| (1, 11)* | <i>abcg abdh acei adfj aefk Aabcdef</i>                          | (1, 3, 3) | (1, 11, 0, 11, 10) |
| (2, 10)* | <i>abcf abdg aceh adei abcdej ABacd</i>                          | (2, 2, 3) | (2, 10, 1, 20, 0)  |
| (3, 9)°  | <i>ABC Aabe Aacf Badg Abcdh ABabcdi</i>                          | (2, 2, 2) | (0, 9, 0, 17, 16)  |
| (5, 7)*  | <i>ABD ACE BCabd BCace BCbcf Aabcg</i>                           | (2, 2, 2) | (0, 7, 0, 35, 6)   |
| (6, 6)*  | <i>ABD ACE BCF ABCabd ABCace ABCbcf</i>                          | (2, 2, 2) | (0, 6, 0, 36, 0)   |
| (7, 5)°  | <i>ABD ACE BCF ABCG Aabd Bace</i>                                | (2, 2, 2) | (0, 5, 0, 29, 4)   |
| (9, 3)*  | <i>ABE ACF ADG BCDH ABCDI BCabc</i>                              | (2, 2, 2) | (0, 3, 0, 27, 3)   |
| (1, 12)* | <i>abcg abdh abei Abcde acfj defk acdefl</i>                     | (1, 3, 3) | (1, 12, 0, 12, 0)  |
| (2, 11)* | <i>abcf abdg acdh abei acej adek ABbcd</i>                       | (2, 2, 3) | (2, 11, 1, 22, 0)  |
| (3, 10)° | <i>ABC Aabe Aacf Bbcg Aadh Abcdi ABabcdj</i>                     | (2, 2, 2) | (0, 10, 0, 10, 17) |
| (6, 7)*  | <i>ABD ACE BCF ABCabd ABCace ABCbcf abcg</i>                     | (2, 2, 2) | (0, 7, 0, 42, 0)   |
| (7, 6)°  | <i>ABD ACE BCF ABCG Aabd Bace Cbcf</i>                           | (2, 2, 2) | (0, 6, 0, 30, 3)   |
| (10, 3)* | <i>ABE ACF BCG ADH BCDI ABCDJ BDabc</i>                          | (2, 2, 2) | (0, 3, 0, 30, 3)   |
| (1, 13)* | <i>abcg abdh acei adej abfk Abcdf ce fl defm</i>                 | (1, 3, 3) | (1, 13, 0, 13, 0)  |
| (2, 12)* | <i>abcf abdg acdh bcdi abej acek adel ABbce</i>                  | (2, 2, 3) | (2, 12, 1, 24, 0)  |
| (3, 11)° | <i>ABC Aabe Aacf Bbcg Aadh Bbdi Bacdj Abcdk</i>                  | (2, 2, 2) | (0, 11, 0, 8, 6)   |
| (7, 7)°  | <i>ABD ACE BCF ABCG Aabd Bace Cbcf ABCabcg</i>                   | (2, 2, 2) | (0, 7, 0, 28, 0)   |
| (11, 3)* | <i>ABE ACF BCG ADH BDI ACDJ BCDK ABCDabc</i>                     | (2, 2, 2) | (0, 3, 0, 33, 3)   |
| (1, 14)* | <i>abcg abdh acei adej abcdek abfl Aacd f aefm abcefn</i>        | (1, 3, 3) | (1, 14, 0, 14, 0)  |
| (2, 13)* | <i>abcf abdg acdh bcdi abej acek bcel adem ABbde</i>             | (2, 2, 3) | (2, 13, 1, 26, 0)  |
| (3, 12)° | <i>ABC Aabe Aacf Bbcg Aadh Bbdi ABacd j Abcdk abcdl</i>          | (2, 2, 2) | (0, 12, 0, 7, 3)   |
| (12, 3)* | <i>ABE ACF BCG ADH BDI ACDJ BCDK ABCDL ABCabc</i>                | (2, 2, 2) | (0, 3, 0, 36, 3)   |
| (1, 15)* | <i>abcg abdh acei adej abcdek abfl Aacd f aefm abcefn abdefo</i> | (1, 3, 3) | (1, 15, 0, 15, 0)  |
| (2, 14)* | <i>abcf abdg acdh bcdi abej acek bcel adem bden ABcde</i>        | (2, 2, 3) | (2, 14, 1, 28, 0)  |
| (3, 13)° | <i>ABC Aabe Aacf Abcg Babch Aadi Abdj Babdk Bcdl acdm</i>        | (2, 2, 2) | (0, 13, 0, 6, 2)   |
| (13, 3)* | <i>ABE ACF BCG ABCH ADI BDJ ABDK CDL ACDM BCDabc</i>             | (2, 2, 2) | (0, 3, 0, 39, 3)   |

tight to simultaneously have a crossing structure and good estimating capability for important effects. It may suggest that a single-array approach should be considered instead; see the good single arrays with  $l_c = 3$  and  $l_n = 6$  reported in Table 3.

Some of the designs reported in Tables 1 and 2 were also obtained by Hedayat and Stufken (1999) and Bingham and Sitter (2003). Nonetheless, they are included in the tables for completeness.

Table 3. The 16-, 32-, and 64-run optimal ESAs

| Array         | Generators   | Clear effects     |
|---------------|--|-------------------|
| <b>16-run</b> |  |                   |
| (2, 4)        | <i>abc</i> <i>ABad</i>   | (2, 1, 0, 4, 2)   |
| (2, 5)        | <i>abd ace</i> <i>ABbc</i>   | (2, 0, 0, 2, 0)   |
| (2, 6)        | <i>abd ace bcf</i> <i>ABabc</i>  | (2, 0, 0, 0, 0)   |
| <b>32-run</b> |  |                   |
| (4, 5)        | <i>ABac</i> <i>ABbd</i> <i>Aabe</i> <i>BCDab</i>   | (4, 5, 5, 10, 0)  |
| (5, 4)        | <i>abd</i> <i>ACac</i> <i>BDac</i> <i>ABEb</i>   | (5, 1, 0, 7, 2)   |
| (2, 8)        | <i>abd ace bcf</i> <i>abcbg</i> <i>ABah</i>  | (2, 1, 0, 12, 6)  |
| (4, 6)        | <i>abd ace bcf</i> <i>ACabc</i> <i>ABDa</i>  | (4, 0, 0, 8, 0)   |
| (5, 5)        | <i>abd ace</i> <i>ACbc</i> <i>BDbc</i> <i>ABEa</i>   | (5, 0, 0, 4, 0)   |
| (6, 4)        | <i>abc</i> <i>ABDa</i> <i>ACEa</i> <i>BCFa</i> <i>ABCbd</i>  | (6, 1, 0, 6, 2)   |
| (2, 9)        | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>ABbd</i>   | (2, 0, 0, 10, 0)  |
| (3, 8)        | <i>abd ace bcf</i> <i>Aabcg</i> <i>Babch</i> <i>ABCa</i>   | (3, 2, 0, 4, 0)   |
| (4, 7)        | <i>abd ace bcf</i> <i>ACabc</i> <i>Babcbg</i> <i>ABDa</i>  | (4, 1, 0, 4, 0)   |
| (5, 6)        | <i>abd ace bcf</i> <i>ACabc</i> <i>BDabc</i> <i>ABEa</i>   | (5, 0, 0, 4, 0)   |
| (6, 5)        | <i>abd ace</i> <i>ACbc</i> <i>BDbc</i> <i>ABEa</i> <i>ABFabc</i>   | (6, 0, 0, 0, 0)   |
| (7, 4)        | <i>ABac</i> <i>ACDa</i> <i>BCEa</i> <i>ABCF</i> <i>Aabd</i> <i>BGab</i>  | (7, 4, 0, 0, 0)   |
| (8, 3)        | <i>ABDa</i> <i>ACEa</i> <i>BCFa</i> <i>ABCG</i> <i>Aabc</i> <i>BHab</i>  | (8, 3, 0, 0, 0)   |
| (2, 10)       | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>bdj</i> <i>ABcd</i>  | (2, 0, 0, 8, 0)   |
| (3, 9)        | <i>abf</i> <i>acg</i> <i>adh</i> <i>Abcd</i> <i>aei</i> <i>Bbce</i> <i>Cbde</i>  | (3, 0, 0, 3, 0)   |
| (4, 8)        | <i>abd ace bcf</i> <i>ACabc</i> <i>Babcbg</i> <i>ABDa</i> <i>ABbch</i>   | (4, 2, 0, 0, 0)   |
| (5, 7)        | <i>abd ace bcf</i> <i>ACabc</i> <i>BDabc</i> <i>ABEa</i> <i>ABbcb</i>  | (5, 1, 0, 0, 0)   |
| (6, 6)        | <i>abd ace bcf</i> <i>ACabc</i> <i>BDabc</i> <i>ABEa</i> <i>ABFbc</i>  | (6, 0, 0, 0, 0)   |
| (8, 4)        | <i>ACab</i> <i>Dabc</i> <i>Aacd</i> <i>AEbc</i> <i>BFab</i> <i>ABGa</i> <i>ABHb</i>  | (8, 4, 0, 0, 0)   |
| (2, 11)       | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>bdj</i> <i>abdk</i> <i>ABcd</i>  | (2, 0, 0, 6, 0)   |
| (3, 10)       | <i>abe acf</i> <i>bcg</i> <i>adh</i> <i>bdi</i> <i>ABabc</i> <i>Aabdj</i> <i>ACcd</i>                                      | (3, 1, 0, 0, 0)   |
| (4, 9)        | <i>abf</i> <i>acg</i> <i>adh</i> <i>Abcd</i> <i>aei</i> <i>Bbce</i> <i>Cbde</i> <i>Dcde</i>                                | (4, 0, 0, 4, 0)   |
| (2, 12)       | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>bdj</i> <i>abdk</i> <i>cdl</i> <i>ABacd</i>                            | (2, 0, 0, 4, 0)   |
| (3, 11)       | <i>abe acf</i> <i>bcg</i> <i>adh</i> <i>bdi</i> <i>acdj</i> <i>ABabc</i> <i>ACabd</i> <i>Acdk</i>                          | (3, 1, 0, 0, 0)   |
| (2, 13)       | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>bdj</i> <i>abdk</i> <i>cdl</i> <i>acdm</i> <i>ABbcd</i>                | (2, 0, 0, 2, 0)   |
| (3, 12)       | <i>abe acf</i> <i>bcg</i> <i>adh</i> <i>bdi</i> <i>acdj</i> <i>bcdk</i> <i>ABabc</i> <i>ACabd</i> <i>Acdl</i>              | (3, 1, 0, 0, 0)   |
| (2, 14)       | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>bdj</i> <i>abdk</i> <i>cdl</i> <i>acdm</i> <i>bcdn</i> <i>ABabcd</i>   | (2, 0, 0, 0, 0)   |
| (3, 13)       | <i>abe acf</i> <i>bcg</i> <i>adh</i> <i>bdi</i> <i>acdj</i> <i>bcdk</i> <i>abcdl</i> <i>ABabc</i> <i>ACabd</i> <i>Acdm</i> | (3, 1, 0, 0, 0)   |
| <b>64-run</b> |  |                   |
| (4, 8)        | <i>abe acf</i> <i>bcg</i> <i>Aabch</i> <i>ACad</i> <i>BDabcd</i>   | (4, 2, 5, 20, 4)  |
| (8, 4)        | <i>Cabc</i> <i>Dabd</i> <i>AEacd</i> <i>BFacd</i> <i>ABGab</i> <i>ABHbcd</i>   | (8, 4, 12, 24, 0) |
| (4, 9)        | <i>abe acf</i> <i>bcg</i> <i>adh</i> <i>Aabci</i> <i>ACbd</i> <i>ABDacd</i>  | (4, 1, 5, 20, 2)  |
| (5, 8)        | <i>abd ace bcf</i> <i>abcbg</i> <i>ABDa</i> <i>ACEb</i> <i>BCch</i>  | (5, 1, 2, 26, 5)  |
| (8, 5)        | <i>Aabd</i> <i>Aace</i> <i>ADbc</i> <i>BEabc</i> <i>CFabc</i> <i>ABCG</i> <i>ABCHabc</i>                                   | (8, 5, 6, 30, 0)  |
| (9, 4)        | <i>ABCE</i> <i>ABDF</i> <i>ACDG</i> <i>ABHa</i> <i>BCDbc</i> <i>Aabd</i> <i>ABCDIab</i>                                    | (9, 4, 6, 16, 0)  |
| (4, 10)       | <i>abf</i> <i>acg</i> <i>bch</i> <i>adi</i> <i>bdj</i> <i>Babce</i> <i>Cabde</i> <i>ADcde</i>                              | (4, 1, 5, 20, 0)  |
| (5, 9)        | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>ACbd</i> <i>BDabd</i> <i>ABEc</i>                                      | (5, 0, 2, 23, 0)  |
| (6, 8)        | <i>abd ace bcf</i> <i>abcbg</i> <i>ABDa</i> <i>ACEb</i> <i>BCFc</i> <i>ABCabch</i>   | (6, 1, 0, 24, 4)  |
| (8, 6)        | <i>abd ace bcf</i> <i>ADabc</i> <i>ABEa</i> <i>ACFa</i> <i>BCGabc</i> <i>ABCHbc</i>  | (8, 0, 0, 24, 0)  |
| (9, 5)        | <i>abd ace</i> <i>ADbc</i> <i>BEbc</i> <i>ABFa</i> <i>ACGa</i> <i>BCHabc</i> <i>ABCIBC</i>                                 | (9, 0, 0, 20, 0)  |
| (10, 4)       | <i>ABEa</i> <i>ACFa</i> <i>BDGa</i> <i>CDHa</i> <i>ABCDIa</i> <i>Aabc</i> <i>BCabd</i> <i>ABCJb</i>                        | (10, 4, 4, 12, 0) |
| (4, 11)       | <i>abe acf</i> <i>bcg</i> <i>adh</i> <i>bdi</i> <i>acdj</i> <i>ACabc</i> <i>Aabdk</i> <i>ABDcd</i>                         | (4, 1, 5, 22, 0)  |
| (5, 10)       | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>bdj</i> <i>ACcd</i> <i>BDacd</i> <i>ABEb</i>                           | (5, 0, 2, 20, 0)  |
| (6, 9)        | <i>abe acf</i> <i>bcg</i> <i>abch</i> <i>adi</i> <i>ACbd</i> <i>BDbd</i> <i>ABEc</i> <i>ABFabcd</i>                        | (6, 0, 0, 20, 0)  |

4. OPTIMAL SINGLE ARRAYS

From the discussion in Section 3.1, we know that the smallest COA or cross array with  $l_c$  control-factors and  $l_n$  noise factors requires  $2^{\lceil \log_2(l_c+1) \rceil + \lceil \log_2(l_n+1) \rceil}$  runs. When both  $l_c$  and  $l_n$  increase, COAs become too large to be feasible in practice; for example, there does not exist a 32-run COA with  $l_c = 4$  and  $l_n = 6$ . If an experimenter can afford to conduct only a 32-run experiment, then he or she should consider single arrays. For given  $l_c, l_n$ , and run size  $2^k$ , a single array is formally a

$2^{(l_c+l_n)-m}$  design with  $l_c$  columns assigned to the control factors and the remaining  $l_n$  columns assigned to the noise factors, where  $m = l_c + l_n - k$  (Wu and Zhu 2003). The necessary and sufficient condition for the existence of a single array is  $l_c + l_n \leq 2^k - 1$ . As an example, the single array generated by the independent defining words *abd, ace, bcf, ACabc,* and *ABDa* might be considered for the experiment with four control factors (*A, B, C, D*) and six noise factors (*a, b, c, d, e, f*).

Table 3. (Continued)

| Array   | Generators   | Clear effects     |
|---------|--|-------------------|
| 64-run  |  |                   |
| (7, 8)  | <i>abe acf adg Cbcd ADbc Abdh BEabcd ABFb ABGacd</i>         | (7, 1, 0, 14, 1)  |
| (8, 7)  | <i>abe acf adg Cbcd ADbc AEbd BFabcd ABGb ABHacd</i>         | (8, 0, 0, 15, 0)  |
| (9, 6)  | <i>abd ace bcf ADabc BEabc ABFc ACGa BCHb ABCIab</i>         | (9, 0, 4, 12, 0)  |
| (10, 5) | <i>abd ace ADbc BEbc ABFa CGbc ACHa BCIabc ABCJ</i>          | (10, 0, 0, 16, 0) |
| (11, 4) | <i>abd ADac BEac ABFb CGac ACHb BClabc ABCJa ABCKc</i>       | (11, 1, 0, 8, 2)  |
| (4, 12) | <i>abe acf bcg adh bdi acdj bcdk ACabc AabdI ABDcd</i>       | (4, 1, 5, 24, 0)  |
| (5, 11) | <i>abe acf bcg abch adi bdj abdk ACcd BDacd ABEb</i>         | (5, 0, 2, 17, 0)  |
| (6, 10) | <i>abe acf bcg abch adi bdj ACcd BDcd ABEa ABFbcd</i>        | (6, 0, 0, 14, 0)  |
| (7, 9)  | <i>abf acg adh Bbcd aei Cbce Dbde AEbc AFabde AGacde</i>     | (7, 0, 0, 10, 0)  |
| (8, 8)  | <i>abe acf adg Cbcd Abch ADbd AEacd BFabcd ABGa ABHbcd</i>   | (8, 1, 0, 14, 1)  |
| (9, 7)  | <i>abe acf bcg Cabcd ADabc AEad BFabc BGad ABHb ABlabd</i>   | (9, 1, 0, 8, 0)   |
| (10, 6) | <i>abd ace bcf ADabc BEabc ABFa CGabc ACHb BCIC ABCJ</i>     | (10, 0, 3, 12, 0) |
| (11, 5) | <i>abd ace ADbc BEbc ABFa CGbc ACHa BClabc ABCJ ABCKbc</i>   | (11, 0, 0, 12, 0) |
| (12, 4) | <i>ABCE ABDF ACDG BCDH ABla ACJa ADKa ABbc CDabd ABCDLab</i> | (12, 4, 2, 6, 0)  |

COAs are special cases of single arrays in that they have crossing structures. Based on the number of control factors ( $l_c$ ), the number of noise factors ( $l_n$ ), and the run size ( $2^k$ ), we propose classifying single arrays into two categories: *prodigal single arrays* (PSAs) and *economical single arrays* (ESAs). A single array with  $l_c$ ,  $l_n$ , and run size  $2^k$  is a PSA if  $(l_c, l_n) \in \mathcal{S}(2^k)$  and an ESA if  $(l_c, l_n) \notin \mathcal{S}(2^k)$ .

We know that  $2^k$ -run single arrays exist for  $l_c$  control factors and  $l_n$  noise factors if  $l_c + l_n \leq 2^k - 1$ . For  $2^k$ -run single arrays, the total degrees of freedom are fixed at  $2^k$ , but  $l_c$  and  $l_n$  can vary. Heuristically, when  $l_c + l_n$  is small, the number of lower-order effects (e.g., the control main effects and the control-by-noise 2fi's) is small, so it may be possible to construct a single array in such a way that all of the lower-order effects are clearly estimable. In this case, we claim that the single array has large relative estimation capacity with respect to  $l_c$  and  $l_n$ . When  $l_c + l_n$  is large, the number of lower-order effects is large, so it may not be possible to have a single array in which all of the lower-order effects are clearly estimable. In this case, we claim that the single array has small relative estimation capacity with respect to  $l_c$  and  $l_n$ . One purpose of classifying single arrays into PSAs and ESAs is to distinguish single arrays with large relative estimation capacities (i.e., PSAs) from those with small relative estimation capacities (i.e., ESAs). As explained later, different criteria must be used for selecting optimal PSAs and ESAs. Another purpose of this classification is to facilitate a fair comparison between cross-arrays and single arrays. In the parameter design literature there is much discussion regarding the advantages and disadvantages of single arrays versus cross-arrays; however, no general conclusions have been reached. We suggest that cross-arrays or COAs are comparable only with PSAs and are not directly comparable with ESAs. This greatly clarifies the discussion and leads to various useful criteria for choosing COAs, PSAs, and ESAs. For the rest of this section, we propose optimality criteria for single arrays in general (Sec. 4.1), discuss the selection of optimal ESAs (Sec. 4.2), and discuss the selection of optimal PSAs (Sec. 4.3).

#### 4.1 Optimality Criteria for Single Arrays

**4.1.1  $W_s$  Aberration.** The wordtype patterns  $A_{i,j}$  of single arrays are not subject to the restriction  $A_{i,1} = A_{i,2} = 0$ .

The arguments used to arrange  $A_{i,j}$  with  $j \geq 3$  into  $W_c$  in Section 3.2 remain valid for single arrays. To obtain a complete sequence of wordtype patterns for single arrays, we need a proper way to insert  $A_{i,j}$  with  $j = 1, 2$  into  $W_c$ . First, consider  $A_{2,1}$  and  $A_{1,2}$  or, equivalently,  $CCn$  and  $Cnn$ . The worst alias relations induced by  $CCn$  and  $Cnn$  are  $C = Cn$  and  $Cn = n$ . Because  $C$  and  $Cn$  are of primary importance for parameter design,  $C = Cn$  is considered more severe than  $Cn = n$ . Thus  $CCn \triangleleft Cnn$ . Clearly, both  $CCn$  and  $Cnn$  are more severe in terms of aliasing severity than  $CCC$  and  $nnn$ . Thus, we have that  $CCn \triangleleft Cnn \triangleleft CCC \triangleleft nnn$ , or  $A_{2,1} \triangleleft A_{1,2} \triangleleft A_{3,0} \triangleleft A_{0,3}$ . Next, we consider  $A_{4,1}$  and  $A_{4,2}$ , or  $CCCn$  and  $CCnn$ . The worst aliasing relation induced by  $CCCn$  is  $CC = Cn$ , and the worst by  $CCnn$  is  $Cn = Cn$ . Both  $CC$  and  $Cn$  are effects of order 2, but  $Cn$  plays a more crucial role in parameter design than  $CC$ . Thus  $Cn = Cn$  is considered more severe than  $CC = Cn$ , which implies that  $CCnn \triangleleft CCCn$ . In a similar way, we have  $CCCn \triangleleft Cnnn$ . Thus we have that  $CCnn \triangleleft CCCn \triangleleft Cnnn \triangleleft CCCC \triangleleft nnnn$ , or  $A_{2,2} \triangleleft A_{3,1} \triangleleft A_{1,3} \triangleleft A_{4,0} \triangleleft A_{0,4}$ . We can further compare defining words with length  $> 4$ . This leads to the same scheme as stated in (3) for ranking two different wordtype patterns,  $A_{i_1,j_1}$  and  $A_{i_2,j_2}$ , in terms of aliasing severity. We denote the resulting sequence  $W_s$ ,

$$W_s = (A_{2,1}, A_{1,2}, A_{3,0}, A_{0,3}, A_{2,2}, A_{3,1}, A_{1,3}, A_{4,0}, A_{0,4}, A_{3,2}, A_{2,3}, A_{4,1}, A_{1,4}, A_{5,0}, A_{0,5}, \dots). \quad (4)$$

Note that  $W_c$  is a subsequence of  $W_s$ .

The hierarchical ordering principle states that (a) effects of lower order are more important than those of higher order, and (b) effects of the same order are equally important. In  $W_s$ , (a) is preserved, but (b) does not hold. In fact, effects with the same order are further distinguished according to their relative importance for parameter design. Based on  $W_s$ ,  $W_s$  aberration and the minimum  $W_s$  aberration criterion for selecting optimal single arrays can be defined in the usual manner. In the derivation of  $W_s$ , we have considered only the aliasing relations implied by the wordtype patterns and have not taken into account the run size of a single array or, to be more precise, the relative estimation capacity of a single array. When the relative estimation capacity of a single array is very limited (e.g., in an economical



single array), it is impossible to guarantee that all of the lower-order effects can be clearly estimated. Thus, in selecting practically useful single arrays, one must prioritize the estimation of important effects versus less important effects. In extreme cases, the estimation of less important effects may need to be compromised entirely.

**4.1.2 Split Wordtype Patterns and  $(W_{sm}, W_{sn})$  Aberration.** The major advantage of using ESAs is run size economy. As mentioned in the end of the previous section, the relative estimation capacities of ESAs are already small. In other words, ESAs usually do not have sufficient capacity to accommodate all of the low-order effects in a balanced way. Nonetheless, the use of ESAs in practice can be justified by the effects sparsity principle, as well as the effects asymmetry inherent in parameter design (Shoemaker et al. 1991). The effects sparsity principle states that only a relatively small number of effects are significant in a factorial experiment (Box and Meyer 1986). Effects asymmetry refers to the fact that control-by-noise 2fi's as well as control main effects are more important than noise main effects and other 2fi's. An ESA that ensures the clear estimation of the important effects (e.g., control main effects and control-by-noise 2fi's) while sacrificing other, less important effects (e.g., noise effects) is still a practically useful experimental plan. To reflect the emphasis on the important effects and the discrimination against the less important effects, we first split  $W_s$  into two separate sequences:

$$W_{sm} = (A_{2.1}, A_{1.2}, A_{3.0}, A_{2.2}, A_{3.1}, A_{1.3}, A_{4.0}, A_{3.2}, A_{2.3}, A_{4.1}, A_{1.4}, A_{5.0}, A_{3.3}, \dots) \quad (5)$$

and

$$W_{sn} = (A_{0.3}, A_{0.4}, A_{0.5}, A_{0.6}, \dots). \quad (6)$$

Note that  $W_{sm}$  consists of  $A_{i,j}$  with  $i \geq 1$ , which involve at least one control factor, whereas  $W_{sn}$  consists of  $A_{i,j}$  with  $i = 0$ , which involve only noise factors. We call  $W_{sm}$  the *mixed wordtype pattern sequence* and  $W_{sn}$  the *noise wordtype pattern sequence*. Next, we append  $W_{sn}$  to the end of  $W_{sm}$  to form the sequence  $(W_{sm}, W_{sn})$ , which is called the *split wordtype pattern sequence*.

Based on this sequence,  $(W_{sm}, W_{sn})$  aberration and the minimum  $(W_{sm}, W_{sn})$  aberration criterion can be defined. Note that in  $(W_{sm}, W_{sn})$ , noise effects are treated as secondary and are sacrificed to ensure increased estimation capacity for control effects and control-by-noise interactions. Thus, both parts (a) and (b) of the hierarchical ordering principle are violated. It is known that defining words that contain only noise factors also can induce aliasing between important effects; for instance,  $nnn$  induces  $Cn = Cnn$ . As discussed in the next section, the split wordtype pattern sequence is one possible compromising scheme aimed at allocating estimation capacity to important effects. It usually does not guarantee the clear estimation of all important effects; however, neither do other schemes. We propose  $(W_{sm}, W_{sn})$  first because it is a systematic scheme that has clear combinatorial structure and often leads to overall good single arrays, especially good ESAs.

**4.1.3 Shifted Wordtype Patterns and  $W_{ss}$  Aberration.** In contrast to  $(W_{sm}, W_{sn})$ , milder compromising schemes can be introduced by shifting  $A_{0,j}$  with  $j \geq 3$  rightward to new positions in  $W_s$  instead of appending all of them at the end of  $W_{sm}$ . In this section we propose one possible shifting scheme. Because the estimation of noise effects is to be compromised, we ignore the aliasing between them in the discussion that follows. We follow the worst-case argument used by Bingham and Sitter (2003). Consider  $A_{0,3}$  or  $nnn$ . The worst alias relation induced by  $nnn$  is  $Cn = Cnn$ , which is similar to  $C_1n = C_2nn$ , the worst aliasing relation induced by  $CCnnn$ . This implies that  $nnn$  and  $CCnnn$  are comparable in terms of aliasing severity. Thus we can have either  $nnn \triangleleft CCnnn$  or  $CCnnn \triangleleft nnn$ . We decide to select  $nnn \triangleleft CCnnn$  and move  $A_{0,3}$  to the position between  $A_{3,2}$  and  $A_{2,3}$ . Next, consider  $A_{0,4}$  or  $nnnn$ . The worst alias relation induced by  $nnnn$  is  $Cnn = Cnn$ , which is similar to  $C_1nn = C_2nn$ , the worst alias relation induced by  $CCnnnn$ . Thus, we shift  $A_{0,4}$  to the position between  $A_{4,2}$  and  $A_{2,4}$ . In general, following the same argument, we have  $A_{i,2} \triangleleft A_{0,i} \triangleleft A_{2,i}$  for  $i \geq 3$ . Shifting all  $A_{0,i}$  rightward as described earlier,  $W_s$  becomes

$$W_{ss} = (A_{2.1}, A_{1.2}, A_{3.0}, A_{2.2}, A_{3.1}, A_{1.3}, A_{4.0}, A_{3.2}, \mathbf{A_{0.3}}, A_{2.3}, A_{4.1}, A_{1.4}, A_{5.0}, A_{3.3}, A_{4.2}, \mathbf{A_{0.4}}, A_{2.4}, A_{5.1}, A_{1.5}, A_{6.0}, A_{4.3}, \dots), \quad (7)$$

with  $A_{0,3}$  and  $A_{0,4}$  highlighted to indicate their new positions.  $W_{ss}$  is called the *shifted wordtype pattern sequence*.

The difference between  $W_{ss}$  and the list of rank-ordered effects given by Bingham and Sitter (2003) is that  $W_{ss}$  further distinguishes wordtype patterns with the same modified wordlength. For example,  $CCn$  and  $Cnn$  have the same modified wordlength, 2.5, but  $CCn \triangleleft Cnn$  in  $W_{ss}$ ; in addition,  $CCC$  and  $CCnn$  have the same modified wordlength, 3, but  $CCC \triangleleft CCnn$  in  $W_{ss}$ . Bingham and Sitter (2003) treated  $nnn$  as equally important as  $CCCC$ ,  $CCCnn$ , and  $CCnnn$ , whereas in  $W_{ss}$ ,  $CCCC \triangleleft CCCnn \triangleleft nnn \triangleleft CCnnn$ . If wordtypes with the same modified wordlength were combined, then  $W_{ss}$  would be reduced to

$$W_{DR} = (A_{2.1} + A_{1.2}, A_{3.0} + A_{2.2}, A_{3.1} + A_{1.3}, A_{4.0} + A_{3.2} + \mathbf{A_{0.3}} + A_{2.3}, A_{4.1} + A_{1.4}, A_{5.0} + A_{3.3} + A_{4.2} + \mathbf{A_{0.4}} + A_{2.4}, A_{5.1} + A_{1.5}, A_{6.0} + A_{4.3}, \dots), \quad (8)$$

which is exactly the sequence proposed by Bingham and Sitter (2003). The  $W_{ss}$  aberration and  $W_{DR}$  aberration can be defined in a standard fashion, as can the minimum  $W_{ss}$  aberration criterion and  $W_{DR}$  aberration criterion. Bingham and Sitter (2003) reported minimum  $W_{DR}$  aberration designs. Because  $W_{DR}$  is a relatively coarse sequence, it may not be able to discriminate among designs that may have different aliasing and structural properties.

*Example 3.* Consider a 32-run experiment involving 2 control factors ( $A, B$ ) and 5 noise factors ( $a, b, c, d, e$ ). Three minimum  $W_{DR}$  aberration arrays were reported in table 4 of Bingham and Sitter (2003):  $d_1 : I = abcd = ABabe = ABCde$ ,  $d_2 : I = abc = ABade = ABbcde$ , and  $d_3 : I = abc = ade = bcde$ . Arrays  $d_1, d_2$ , and  $d_3$  share the same  $W_{DR} = (0, 0, 0, 2, 0, 1)$ , and all are in fact COAs. Their strength vectors are  $T(d_1) = (2, 2, 3)$ ,

$T(d_2) = (2, 2, 2)$ , and  $T(d_3) = (2, 2, 2)$ ;  $d_1$  has greater strength than  $d_2$  and  $d_3$ . All of the arrays guarantee the clear estimation of control main effects and control-by-noise 2fi's. In addition,  $d_1$  guarantees the clear estimation of noise main effects; therefore,  $d_1$  is better overall than the other two arrays. The shifted wordtype pattern sequences of  $d_1$ ,  $d_2$ , and  $d_3$  are  $W_{ss}(d_1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, \dots)$ ,  $W_{ss}(d_2) = (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, \dots)$ , and  $W_{ss}(d_3) = (0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, \dots)$ . Thus,  $d_1$  is the minimum  $W_{ss}$  aberration single array, followed by  $d_2$  and  $d_3$ .

In summary,  $W_{ss}$  is more elaborate than  $W_{DR}$ ; therefore, it is more sensitive in discriminating among different designs. On the other hand,  $W_{DR}$  is more conservative, and the minimum  $W_{DR}$  aberration criterion usually leads to a group of arrays. As such, good arrays will not be missed.

### 4.2 Optimal Economical Single Arrays

Because PSAs and ESAs are fairly different from each other in terms of relative estimation capacities, different optimality criteria should be used to select optimal PSAs and ESAs. In this section we focus on the selection of optimal ESAs. For ESAs, because their relative estimation capacities are already limited, the minimum  $W_{ss}$  aberration and  $(W_{sm}, W_{sn})$  aberration criteria are more appropriate than the minimum  $W_s$  aberration criterion. We have also found that  $W_{ss}$  aberration and  $(W_{sm}, W_{sn})$  aberration often lead to the same optimal ESAs. Thus, the minimum  $(W_{sm}, W_{sn})$  aberration criterion more genuinely reflects the structure of optimal ESAs. In this article we report only the ESAs with minimum  $(W_{sm}, W_{sn})$  aberrations.

Similar to the selection of optimal COAs, we carried out an exhaustive search with isomorphism checking to select ESAs with minimum  $(W_{sm}, W_{sn})$  aberration and obtained all optimal 16-run and 32-run ESAs, as well as optimal 64-run ESAs with up to 16 factors. Due to space limitations, only some of the optimal ESAs are chosen; these are presented in Table 3. Two general rules guided the choice of optimal ESAs. First, Table 3 includes only optimal ESAs with  $l_c + l_n \leq 2^{k-1}$  for  $k = 4, 5$  and with  $l_c + l_n \leq 16$  for  $k = 6$ . Second, if an optimal ESA with  $l_c$  and  $l_n$  is also an ordinary MA  $2^{(l_c+l_n)-m}$  design ( $m = l_c + l_n - k$ ), then it is not listed in Table 3. The complete tables of 16-, 32-, and 64-run optimal ESAs are available from the authors on request.

### 4.3 Optimal Prodigious Single Arrays

Compared with ESAs, PSAs have larger relative estimation capacities for fixed  $l_c$  and  $l_n$ . Thus discrimination against less important effects may not be as necessary in the selection of optimal PSAs as it is in the selection of optimal ESAs. Among the three optimality criteria proposed in Sections 4.1.2 and 4.1.3, it appears that the minimum  $W_s$  aberration criterion is the most suitable for PSAs. For most  $l_c$  and  $l_n$ , the optimal PSAs according to the minimum  $W_s$  aberration criterion are much better than the optimal PSAs according to the other two criteria in terms of strength and the number of clearly estimable effects. However, there are cases in which the latter two criteria select better arrays. These cases usually occur when  $(l_c, l_n)$  are

on the boundary of  $\mathcal{S}(2^k)$  in the sense that there does not exist  $(l'_c, l'_n) \in \mathcal{S}(2^k)$  such that  $l'_c \geq l_c$ ,  $l'_n \geq l_n$ , and  $l'_c + l'_n > l_c + l_n$ . Thus, when constructing tables of optimal PSAs of 16, 32, and 64 runs, we consider all 3 criteria. In the cases where the criteria lead to different optimal PSAs, if one optimal PSA is apparently better than the others, then that is the one selected; if the PSAs are comparable, then we keep all of them in the tables. The obtained optimal PSAs are reported in Table 4 (including 16- and 32-run arrays) and in Table 5 (including 64-run arrays with up to 16 factors).

In both Tables 4 and 5, the first two columns give  $(l_c, l_n)$  and the independent defining words (or generators), the next three columns indicate whether an array is optimal according to  $W_s$ ,  $(W_{sm}, W_{sn})$  or  $W_{ss}$  aberration, and the last two columns indicate whether the array is also a COA and whether it is a MA design with the distinction between control and noise factors neglected. For example, in Table 4, there are two optimal 32-run PSAs for  $l_c = 4$  and  $l_n = 3$ :

| Design | Generators        | $W_s$ | $(W_{sm}, W_{sn})$ | $W_{ss}$ | COA | MA |
|--------|-------------------|-------|--------------------|----------|-----|----|
| (4, 3) | <i>ABCD ABabc</i> | ✓     |                    |          | ✓   | ✓  |
| (4, 3) | <i>abc ABCDa</i>  |       | ✓                  | ✓        |     |    |

We refer to the first array as  $d_1$  and to the second as  $d_2$ . Array  $d_1$  is optimal according to the minimum  $W_s$  aberration criterion and is in fact an optimal COA. On the other hand, array  $d_2$  is optimal according to the minimum  $(W_{sm}, W_{sn})$  aberration and  $W_{ss}$  aberration criteria. Note that for a 32-run parameter design experiment involving 4 control factors and 3 noise factors, the optimal COA coincides with the optimal PSA according to  $W_s$  aberration. But, this is not generally true in other cases, as shown in Tables 4 and 5. It is not difficult to see that  $d_2$  is not a COA. Although using array  $d_2$  guarantees the clear estimation of all control main effects, control-by-control 2fi's, and control-by-noise 2fi's, the noise main effects and noise-by-noise 2fi's are aliased with one another. In  $d_1$ , the noise main effects and noise-by-noise 2fi's are clearly estimable, but the control-by-control 2fi's are aliased with one another. As a single array,  $d_2$  is better than  $d_1$  if the noise 2fi's are assumed to be negligible.

When  $l_c = 3$  and  $l_n = 5$ , there are also two optimal 32-run PSAs reported in Table 4:

|                    | Design                 |                      |
|--------------------|------------------------|----------------------|
|                    | (3, 5)                 | (3, 5)               |
| Generators         | <i>Aabd Aace ABCbc</i> | <i>abd ace ABCbc</i> |
| $W_s$              | ✓                      |                      |
| $(W_{sm}, W_{sn})$ |                        | ✓                    |
| $W_{ss}$           |                        | ✓                    |
| COA                |                        |                      |
| MA                 | ✓                      |                      |

We designate the first array  $d_3$  and the second array  $d_4$ . Array  $d_3$  is optimal according to the minimum  $W_s$  aberration criterion, whereas array  $d_4$  is optimal according to the minimum  $(W_{sm}, W_{sn})$  aberration and  $W_{ss}$  aberration criteria. Neither  $d_3$

Table 4. The 16- and 32-run optimal PSAs with  $(l_c, l_n) \in \mathcal{S}(2^k)$

| Array         | Generators  | $W_s$ | $(W_{sm}, W_{sn})$ | $W_{ss}$ | COA | MA |
|---------------|---|-------|--------------------|----------|-----|----|
| <b>16-run</b> |   |       |                    |          |     |    |
| (1, 4)        | <i>Aabcd</i>  | ✓     |                    |          | ✓   | ✓  |
| (2, 3)        | <i>ABabc</i>  | ✓     |                    | ✓        | ✓   | ✓  |
| (3, 2)        | <i>ABCab</i>  | ✓     | ✓                  | ✓        |     | ✓  |
| (4, 1)        | <i>ABCDa</i>  | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 5)        | <i>abd ace</i>  |       | ✓                  | ✓        | ✓   |    |
| (3, 3)        | <i>ABCa ABbc</i>  | ✓     |                    |          |     | ✓  |
| (3, 3)        | <i>ABCa abc</i>   |       | ✓                  | ✓        |     |    |
| (5, 1)        | <i>ABDa ACEa</i>  | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 6)        | <i>abd ace bcf</i>  |       | ✓                  | ✓        | ✓   |    |
| (6, 1)        | <i>ABDa ACEa BCFa</i>   | ✓     | ✓                  | ✓        | ✓   | ✓  |
| (1, 7)        | <i>abd ace bcf abcg</i>   |       | ✓                  | ✓        | ✓   |    |
| (7, 1)        | <i>ABDa ACEa BCFa ABCG</i>                                      | ✓     | ✓                  | ✓        |     | ✓  |
| <b>32-run</b> |   |       |                    |          |     |    |
| (1, 5)        | <i>Aabcde</i>   | ✓     |                    |          | ✓   | ✓  |
| (2, 4)        | <i>ABabcd</i>   | ✓     |                    | ✓        | ✓   | ✓  |
| (3, 3)        | <i>ABCabc</i>   | ✓     |                    | ✓        | ✓   | ✓  |
| (4, 2)        | <i>ABCDab</i>   | ✓     | ✓                  | ✓        |     | ✓  |
| (5, 1)        | <i>ABCDEa</i>   | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 6)        | <i>abce Aabdf</i>   | ✓     |                    |          | ✓   | ✓  |
| (2, 5)        | <i>abcd ABabe</i>   | ✓     |                    | ✓        | ✓   | ✓  |
| (3, 4)        | <i>abcd ABCab</i>   | ✓     |                    |          | ✓   | ✓  |
| (4, 3)        | <i>ABCD ABabc</i>   | ✓     |                    |          | ✓   | ✓  |
| (4, 3)        | <i>abc ABCDa</i>  |       | ✓                  | ✓        |     |    |
| (5, 2)        | <i>ABCD ABEab</i>   | ✓     | ✓                  | ✓        |     | ✓  |
| (6, 1)        | <i>ABCE ABDFa</i>   | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 7)        | <i>abce abdf Aacdg</i>  | ✓     |                    |          | ✓   | ✓  |
| (2, 6)        | <i>abce abdf ABacd</i>  | ✓     |                    | ✓        | ✓   | ✓  |
| (3, 5)        | <i>Aabd Ace ABCbc</i>   | ✓     |                    |          |     | ✓  |
| (3, 5)        | <i>abd ace ABCbc</i>  |       | ✓                  | ✓        |     | ✓  |
| (5, 3)        | <i>ABDa ACEa BCabc</i>  | ✓     |                    |          |     | ✓  |
| (6, 2)        | <i>ABCE ABDF ACDab</i>  | ✓     | ✓                  | ✓        |     | ✓  |
| (7, 1)        | <i>ABCE ABDF ACDGa</i>  | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 8)        | <i>abcf abdg abeh Aacde</i>                                     | ✓     |                    |          | ✓   | ✓  |
| (2, 7)        | <i>abce abdf acdg ABbcd</i>                                     | ✓     |                    | ✓        | ✓   |    |
| (3, 6)        | <i>Aabd Ace Abcf BCabc</i>                                      | ✓     |                    |          |     |    |
| (3, 6)        | <i>abd ace bcf ABCabc</i>                                       |       | ✓                  | ✓        |     |    |
| (6, 3)        | <i>ABDa ACEa BCFa ABCbc</i>                                     | ✓     |                    |          |     | ✓  |
| (7, 2)        | <i>ABCE ABDF ACDG BCDab</i>                                     | ✓     | ✓                  | ✓        |     | ✓  |
| (8, 1)        | <i>ABCF ABDG ABEH ACDEa</i>                                     | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 9)        | <i>abe acf adg bcdh abcdi</i>                                   |       | ✓                  | ✓        | ✓   |    |
| (3, 7)        | <i>abd ace bcf abcg ABCa</i>                                    |       | ✓                  | ✓        |     |    |
| (7, 3)        | <i>abc ABDa ACEa BCFa ABCGb</i>                                 |       | ✓                  | ✓        |     |    |
| (9, 1)        | <i>ABEa ACFa ADGa BCDHa ABCDI</i>                               | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 10)       | <i>abe acf bcg adh bcdi abcdj</i>                               |       | ✓                  | ✓        | ✓   |    |
| (10, 1)       | <i>ABCE ABDF ACDG BCDH ABla ACJa</i>                            | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 11)       | <i>abe acf bcg adh bdi acdj bcdk</i>                            |       | ✓                  | ✓        | ✓   |    |
| (11, 1)       | <i>ABCE ABDF ACDG BCDH ABla ACJa ADKa</i>                       | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 12)       | <i>abe acf bcg adh bdi acdj bcdk abcdl</i>                      |       | ✓                  | ✓        | ✓   |    |
| (12, 1)       | <i>ABCF ABDG ACDH BCDI ABEJ ACEK BCEL ADEa</i>                  | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 13)       | <i>abe acf bcg abch adi bdj abdk cdl acdm</i>                   |       | ✓                  | ✓        | ✓   |    |
| (13, 1)       | <i>ABEa ACFa BCGa ABCH ADIa BDJa ABDK CDLa ACDM ACDM</i>        | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 14)       | <i>abe acf bcg abch adi bdj abdk cdl acdm bcdn</i>              |       | ✓                  | ✓        | ✓   |    |
| (14, 1)       | <i>ABEa ACFa BCGa ABCH ADIa BDJa ABDK CDLa ACDM BCDN</i>        | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 15)       | <i>abe acf bcg abch adi bdj abdk cdl acdm bcdn abedo</i>        |       | ✓                  | ✓        | ✓   |    |
| (15, 1)       | <i>ABEa ACFa BCGa ABCH ADIa BDJa ABDK CDLa ACDM BCDN ABCDOa</i> | ✓     | ✓                  | ✓        |     | ✓  |

Table 5. The 64-run optimal PSAs with  $(l_c, l_n) \in \mathcal{S}(2^k)$

| Array   | Generators                                   | $W_s$ | $(W_{sm}, W_{sn})$ | $W_{ss}$ | COA | MA |
|---------|--|-------|--------------------|----------|-----|----|
| (1, 6)  | <i>Aabcdef</i>                               | ✓     |                    |          | ✓   | ✓  |
| (2, 5)  | <i>ABabcde</i>                               | ✓     |                    | ✓        | ✓   | ✓  |
| (3, 4)  | <i>ABCabcd</i>                               | ✓     |                    | ✓        | ✓   | ✓  |
| (4, 3)  | <i>ABCDabc</i>                               | ✓     |                    | ✓        | ✓   | ✓  |
| (5, 2)  | <i>ABCDEab</i>                               | ✓     | ✓                  | ✓        |     | ✓  |
| (6, 1)  | <i>ABCDEFa</i>                               | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 7)  | <i>Aabcf abdeg</i>                           | ✓     |                    |          | ✓   | ✓  |
| (2, 6)  | <i>Aabce Babdf</i>                           | ✓     |                    |          | ✓   | ✓  |
| (3, 5)  | <i>ABabd Cabce</i>                           | ✓     |                    |          | ✓   | ✓  |
| (4, 4)  | <i>Aabcd ABCDa</i>                           | ✓     |                    |          | ✓   | ✓  |
| (5, 3)  | <i>ABabc ABCDE</i>                           | ✓     |                    | ✓        | ✓   | ✓  |
| (6, 2)  | <i>ABCEa ABDFb</i>                           | ✓     | ✓                  | ✓        |     | ✓  |
| (7, 1)  | <i>ABCFa ABDEG</i>                           | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 8)  | <i>abcf Aabdg acdeh</i>                      | ✓     |                    |          | ✓   | ✓  |
| (2, 7)  | <i>abce Aabdf Bacdg</i>                      | ✓     |                    |          | ✓   | ✓  |
| (3, 6)  | <i>abce ACabd Bacdf</i>                      | ✓     |                    |          | ✓   | ✓  |
| (4, 5)  | <i>abce ACabd ABCDbc</i>                     | ✓     |                    | ✓        |     | ✓  |
| (5, 4)  | <i>abcd ABDab ACEac</i>                      | ✓     |                    |          |     | ✓  |
| (6, 3)  | <i>ABCE ABabc ACDFa</i>                      | ✓     |                    |          |     | ✓  |
| (6, 3)  | <i>abc ABCEa ABDFb</i>                       |       | ✓                  | ✓        |     |    |
| (7, 2)  | <i>ABCE ABDFa ACDGb</i>                      | ✓     | ✓                  | ✓        |     | ✓  |
| (8, 1)  | <i>ABCF ABDGa ACDEH</i>                      | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 9)  | <i>abcf Aabdg Aabeh acdei</i>                | ✓     |                    |          | ✓   | ✓  |
| (2, 8)  | <i>abcf Aabdg Aabeh Bacde</i>                | ✓     |                    |          | ✓   | ✓  |
| (3, 7)  | <i>abcf abdg Bacde ACabe</i>                 | ✓     |                    |          | ✓   | ✓  |
| (4, 6)  | <i>abce abdf ACacd BDbcd</i>                 | ✓     |                    | ✓        |     |    |
| (5, 5)  | <i>abce ACabd BDabd ABEac</i>                | ✓     |                    |          |     | ✓  |
| (5, 5)  | <i>abd ace ABDbc ACEabc</i>                  |       | ✓                  | ✓        |     |    |
| (6, 4)  | <i>abcd ABDab ACEab BCFac</i>                | ✓     |                    |          |     | ✓  |
| (7, 3)  | <i>ABCE ABDF ACDGa ABabc</i>                 | ✓     |                    |          |     |    |
| (7, 3)  | <i>abc ABCE ABDFa ACDGb</i>                  |       | ✓                  | ✓        |     |    |
| (8, 2)  | <i>ABCF ABDGa ABEHa ACDEb</i>                | ✓     | ✓                  | ✓        |     | ✓  |
| (9, 1)  | <i>ABCF ABDGa ABEHa ACDEI</i>                | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 10) | <i>abcg abdh acdei acdfj Aabef</i>           | ✓     |                    |          | ✓   | ✓  |
| (2, 9)  | <i>abcf abdg abeh acdei ABbcde</i>           | ✓     |                    | ✓        | ✓   |    |
| (3, 8)  | <i>abcf abdg abeh Bacde ACbcde</i>           | ✓     |                    |          |     |    |
| (4, 7)  | <i>abce abdf ACacd BDacd ABabg</i>           | ✓     |                    |          | ✓   | ✓  |
| (4, 7)  | <i>abe acf adg ACbcd BDabcd</i>              |       |                    | ✓        |     |    |
| (5, 6)  | <i>abce abdf ACacd BDacd ABEab</i>           | ✓     |                    | ✓        |     |    |
| (6, 5)  | <i>ABCE ABDF ACDac ACDbd ABabe</i>           | ✓     |                    |          |     | ✓  |
| (6, 5)  | <i>abd ace ABDbc ACEbc BCFabc</i>            |       | ✓                  | ✓        |     |    |
| (7, 4)  | <i>ABCE ABDF ACDac ACDbd ABGab</i>           | ✓     |                    |          |     | ✓  |
| (8, 3)  | <i>ABCF ABDG ABEH ACDEb BCDEac</i>           | ✓     |                    |          |     |    |
| (8, 3)  | <i>abc ABCE ABDFa ACDGb BCDHab</i>           |       | ✓                  | ✓        |     |    |
| (9, 2)  | <i>ABCF ABDG ABEH ACDEI BCDEab</i>           | ✓     | ✓                  | ✓        |     |    |
| (10, 1) | <i>ABCG ABDH ACDEI ADFJ ABEFa</i>            | ✓     | ✓                  | ✓        |     | ✓  |
| (1, 11) | <i>abcg abdh acei adfj aefk Aabcdef</i>      | ✓     |                    |          | ✓   |    |
| (2, 10) | <i>abcf abdg aceh adei abcdej ABacd</i>      | ✓     |                    |          | ✓   | ✓  |
| (2, 10) | <i>abcf abdg acdh bcdi abej ABace</i>        |       |                    | ✓        | ✓   |    |
| (3, 9)  | <i>abe acf bcg adh bcdi ABCabcd</i>          |       |                    | ✓        |     |    |
| (3, 9)  | <i>abe acf bcg abch adi ABCabd</i>           |       | ✓                  |          |     |    |
| (5, 7)  | <i>ABCE ABDc ACDad ACDbe ABabf BCDabg</i>    | ✓     |                    |          |     | ✓  |
| (5, 7)  | <i>abe acf adg ACbcd BDabcd ABEb</i>         |       |                    | ✓        |     |    |
| (5, 7)  | <i>abd ace bcf abcg ABDa ACEb</i>            |       | ✓                  |          |     |    |
| (6, 6)  | <i>abce abdf ACacd BDacd ABEab ABFbcd</i>    | ✓     |                    | ✓        |     | ✓  |
| (7, 5)  | <i>abd ace ABDbc ACEbc BCFbc ABCGa</i>       |       | ✓                  | ✓        |     |    |
| (9, 3)  | <i>ABCG ABDH ABEI BCDEa ACFb DEFc</i>        | ✓     |                    |          |     |    |
| (9, 3)  | <i>abc ABEa ACFa ADGb BCDHa ABCDIab</i>      |       | ✓                  | ✓        |     |    |
| (10, 2) | <i>ABCF ABDG ACEH ADEI ABCDEJ ACDab</i>      | ✓     | ✓                  | ✓        |     |    |
| (11, 1) | <i>ABCG ABDH ACEI ADFJ AEFK ABCDEFa</i>      | ✓     | ✓                  | ✓        |     |    |
| (1, 12) | <i>abcg abdh abei Abcde acfj defk acdefl</i> | ✓     |                    |          | ✓   |    |
| (2, 11) | <i>abcf abdg acdh abei acej adek ABbcd</i>   | ✓     |                    |          | ✓   |    |

Table 5. (Continued)

| Array   | Generators  | $W_s$ | $(W_{sm}, W_{sn})$ | $W_{ss}$ | COA | MA |
|---------|---|-------|--------------------|----------|-----|----|
| (3, 10) | <i>abe acf bcg adh bdi acdj ABCbcd</i>                          |       |                    | ✓        |     |    |
| (3, 10) | <i>abe acf bcg abch adi bdj ABCcd</i>                           |       | ✓                  |          |     |    |
| (6, 7)  | <i>ABDa ACEa BCFa ABCbd ABCce abcf ABCabcg</i>                  | ✓     |                    |          |     |    |
| (7, 6)  | <i>abd ace bcf ABDabc ACEabc BCFabc ABCG</i>                    |       |                    | ✓        |     |    |
| (10, 3) | <i>ABCG ABDH ACEI ADEa ABFJ BCDfB CEFc</i>                      | ✓     |                    |          |     |    |
| (10, 3) | <i>abc ABEa ACFa BCGb ADHb BCDIa ABCDJab</i>                    |       | ✓                  | ✓        |     |    |
| (11, 2) | <i>ABCF ABDG ACDH ABEI ACEJ ADEK BCDab</i>                      | ✓     | ✓                  | ✓        |     |    |
| (12, 1) | <i>ABCG ABDH ABEI BCDEa ACfJ DEFk ACDEFL</i>                    | ✓     | ✓                  | ✓        |     |    |
| (1, 13) | <i>abcg abdh acei adej abfk Abcdf cefl defm</i>                 | ✓     |                    |          | ✓   |    |
| (2, 12) | <i>abcf abdg acdh bcdi abej acek adel ABbce</i>                 | ✓     |                    |          | ✓   |    |
| (3, 11) | <i>abe acf bcg adh bdi acdj bcdk ABCabcd</i>                    |       |                    | ✓        |     |    |
| (11, 3) | <i>ABCF ABDG ACEH ADEI ABCDEb ABJa ACDac AEKa</i>               | ✓     |                    |          |     |    |
| (11, 3) | <i>abc ABEa ACFa BCGb ADHa BDib ACDJb BCDKa</i>                 |       | ✓                  | ✓        |     |    |
| (12, 2) | <i>ABCF ABDG ACDH BCDI ABEJ ACEK ADEL BCEab</i>                 | ✓     | ✓                  | ✓        |     |    |
| (13, 1) | <i>ABCG ABDH ACEI ADEJ ABfK BCDfA CEFL DEFM</i>                 | ✓     | ✓                  | ✓        |     |    |
| (1, 14) | <i>abcg abdh acei adej abcdek abfl Aacdf aefm abcefn</i>        | ✓     |                    |          | ✓   |    |
| (2, 13) | <i>abcf abdg acdh bcdi abej acek bcel adem ABbde</i>            | ✓     |                    | ✓        | ✓   |    |
| (3, 12) | <i>abe acf bcg abch adi bdj abdk cdl ABCacd</i>                 |       | ✓                  | ✓        |     |    |
| (12, 3) | <i>ABCG ABDH ACdI BCDJ ABEK ACEa ABfL ADFb ABCDEFc</i>          | ✓     |                    |          |     |    |
| (12, 3) | <i>abc ABEa ACFa BCGb ADHa BDib ACDJab BCDKa ABCDL</i>          |       | ✓                  | ✓        |     |    |
| (13, 2) | <i>ABCF ABDG ACDH BCDI ABEJ ACEK BCEL ADEM BDEab</i>            | ✓     | ✓                  | ✓        |     |    |
| (14, 1) | <i>ABCG ABDH ACEI ADEJ ABCDEK ABfL ACDfA AEFM ABCEFN</i>        | ✓     | ✓                  | ✓        |     |    |
| (1, 15) | <i>abcg abdh acei adej abcdek abfl Aacdf aefm abcefn abdefo</i> | ✓     |                    |          | ✓   |    |
| (2, 14) | <i>abcf abdg acdh bcdi abej acek bcel adem bden ABcde</i>       | ✓     |                    | ✓        | ✓   |    |
| (3, 13) | <i>abe acf bcg abch adi bdj abdk cdl acdm ABCbcd</i>            |       | ✓                  | ✓        |     |    |
| (13, 3) | <i>ABCF ABDG ACDH BCDI ABEJ ACEK ADEb ABLa ACMa ABCDEac</i>     | ✓     |                    |          |     |    |
| (13, 3) | <i>abc ABEa ACFa BCGa ABCHb ADIa BDJa ABDKb CDLb ACDM</i>       |       | ✓                  | ✓        |     |    |
| (14, 2) | <i>ABCF ABDG ACDH BCDI ABEJ ACEK BCEL ADEM BDEN CDEab</i>       | ✓     | ✓                  | ✓        |     |    |
| (15, 1) | <i>ABCG ABDH ACEI ADEJ ABCDEK ABfL ACDfA AEFM ABCEFN ABDEFO</i> | ✓     | ✓                  | ✓        |     |    |

nor  $d_4$  is a COA. The optimal 32-run COA for 3 control factors and 4 noise factors given in Table 1, designated  $d_5$ , is generated by *ABC*, *Aabd*, and *Bace*. In terms of the number of clearly estimable effects, both  $d_3$  and  $d_4$  are better than  $d_5$ .

5. CONCLUDING REMARKS

In this article we have suggested that structural constraints and relative estimation capacities should be taken into consideration when selecting optimal plans for parameter design experiments. We have proposed various optimality criteria for selecting useful COAs, ESAs, and PSAs. Using the tables provided in this article, experimenters can consider and compare possible experimental plans that are optimal in one way or another, and choose the one that best fits their experimental constraints, capacity, and goals. For example, suppose that there are four control factors (*A, B, C, D*) and five noise factors (*a, b, c, d, e*) in a parameter design experiment. Because  $(4, 5) \notin \mathcal{S}(2^5)$ , there does not exist a 32-run COA with  $l_c = 4$  and  $l_n = 5$ . The optimal 64-run COA  $d'$ , generated by *ABCD*, *ABabd*, and *ACace*, is given in Table 2. The experimenter also may consider using optimal 64-run PSAs. Table 5 lists an optimal 64-run PSA  $d''$  with  $l_c = 4$  and  $l_n = 5$  that is generated by *abce*, *ACabd*, and *ABCDbc*. If the crossing structure is not crucial in the experiment and analysis, then  $d''$  appears to be a better choice than  $d'$ , because  $d''$  guarantees clear estimation of the control-by-control 2fi's in addition to the main effects and the control-by-noise 2fi's. If the experimenter cannot afford a 64-run design,

then he or she may consider using 32-run ESAs. The optimal 32-run ESA  $d'''$  with  $l_c = 4$  and  $l_n = 5$ , generated by *ABac*, *ABbd*, *Aabe*, and *BCDab*, is given in Table 3. Due to its limited capacity,  $d'''$  does not guarantee clear estimation of all the important effects.

APPENDIX: PROOF OF PROPOSITION 1

Sufficiency

Because  $(l_c, l_n) \in \mathcal{S}(2^k)$ ,  $\lceil \log_2(l_c + 1) \rceil + \lceil \log_2(l_n + 1) \rceil \leq k$ . Let  $k_c = \lceil \log_2(l_c + 1) \rceil$  and  $k_n = \lceil \log_2(l_n + 1) \rceil$ . It is clear that  $l_c \leq 2^{k_c} - 1$  and  $l_n \leq 2^{k_n} - 1$ . Hence two fractional factorial designs, denoted by  $d_1$  and  $d_2$ , can be constructed for the control factors and the noise factors. Crossing  $d_1$  and  $d_2$  gives an COA with run size  $2^{k_c} \times 2^{k_n} \leq 2^k$ .

Necessity

Suppose that  $d$  is a COA with  $l_c$  control factors,  $l_n$  noise factors and  $2^k$  runs. Let  $N_c$  and  $N_n$  be the run sizes of the control array and the noise arrays.  $N_c N_n = 2^k$ . Because both  $N_c$  and  $N_n$  are powers of 2, there exist  $k_c$  and  $k_n$  such that  $k_c + k_n = k$ ,  $N_c = 2^{k_c}$  and  $N_n = 2^{k_n}$ . According to the definition,  $t_c \geq \min(l_c, 2)$  and  $t_n \geq \min(l_n, 2)$ . Hence the control array and the noise arrays are either full factorial designs or have resolutions higher than three. Hence, one has  $l_c \leq 2^{k_c} - 1$  and  $l_n \leq 2^{k_n} - 1$ , which implies that  $(l_c, l_n) \in \mathcal{S}(2^k)$ .

## ACKNOWLEDGMENTS

Yu Zhu's research was supported by National Science Foundation grant DMS-04-05694. The authors thank the editor, the associate editor and the referees for constructive comments and suggestions that helped improve an early version of the article.

[Received April 2004. Revised March 2006.]

## REFERENCES

- Bingham, D., and Sitter, R. R. (2003), "Fractional Factorial Split-Plot Designs for Robust Parameter Experiments," *Technometrics*, 45, 80–89.
- Box, G. E. P., and Meyer, R. D. (1986), "Dispersion Effects From Fractional Designs," *Technometrics*, 28, 19–27; corr. 29, 250.
- Chen, J., Sun, D. X., and Wu, C. F. J. (1993), "A Catalogue of Two-Level and Three-Level Fractional Factorial Designs With Small Runs," *International Statistical Review*, 61, 131–145.
- Easterling, R. G. (1985), Discussion of "OFF-Line Quality Control, Parameter Design, and the Taguchi Method," by R. N. Karkar, *Journal of Quality Technology*, 17, 191–192.
- Fries, A., and Hunter, W. G. (1980), "Minimum Aberration  $2^{k-p}$  Designs," *Technometrics*, 22, 601–608.
- Hedayat, A. S., and Stufken, J. (1999), "Compound Orthogonal Arrays," *Technometrics*, 41, 57–61.
- Karkar, R. N. (1985), "Off-Line Quality Control, Parameter Design, and the Taguchi Method" (with discussion), *Journal of Quality Technology*, 17, 176–209.
- Lucas, J. M. (1989), "Achieving a Robust Process Using Response Surface Methodology," in *Proceedings of the Sesquicentennial Invited Sessions*, American Statistical Association, pp. 579–593.
- Nair, V. N. (1992), "Taguchi's Parameter Design: A Panel Discussion," *Technometrics*, 34, 127–161.
- Rao, C. R. (1947), "Factorial Experiments Derivable From Combinatorial Arrangements of Arrays," *Journal of the Royal Statistical Society*, Supplement, 9, 128–139.
- Rosenbaum, P. R. (1994), "Dispersion Effects From Fractional Factorials in Taguchi's Method of Quality Design," *Journal of the Royal Statistical Society*, Ser. B, 56, 641–652.
- (1996), "Some Useful Compound Dispersion Experiments in Quality Design," *Technometrics*, 38, 354–364.
- (1999), "Blocking in Compound Dispersion Experiments," *Technometrics*, 41, 125–134.
- Shoemaker, A. C., Tsui, K. L., and Wu, C. F. J. (1991), "Economical Experimentation Methods for Robust Design," *Technometrics*, 33, 415–427.
- Steinberg, D. (1996), "Robust Design: Experiments for Improving Quality," in *Handbook of Statistics 13: Design and Analysis of Experiments*, eds. S. Ghosh and C. R. Rao, New York: North-Holland, pp. 199–240.
- Steinberg, D. M., and Burstzyn, D. (1998), "Noise Factors, Dispersion Effects and Robust Design," *Statistica Sinica*, 8, 67–85.
- Taguchi, G. (1986), *Introduction to Quality Engineering: Designing Quality Into Products and Process*, Tokyo: Asian Productivity Organization.
- Vining, G. G., and Myers, R. H. (1990), "Combining Taguchi and Response Surface Philosophies: A Dual-Response Approach," *Journal of Quality Technology*, 22, 38–45.
- Welch, W. J., Yu, T. K., Kang, S. M., and Sacks, J. (1990), "Computer Experiments for Quality Control by Parameter Design," *Journal of Quality Technology*, 22, 15–22.
- Wu, C. F. J., and Chen, Y. Y. (1992), "A Graph-Aided Method for Planning Two-Level Experiments When Certain Interactions Are Important," *Technometrics*, 34, 162–175.
- Wu, C. F. J., and Hamada, M. S. (2000), *Experiments: Planning, Analysis and Parameter Design Optimization*, New York: Wiley.
- Wu, C. F. J., and Zhu, Y. (2003), "Optimal Selection of Single Arrays for Parameter Design Experiments," *Statistica Sinica*, 13, 1179–1199.
- Zhu, Y. (2003), "Structure Function for Aliasing Patterns in  $2^{l-n}$  Design With Multiple Groups of Factors," *The Annals of Statistics*, 31, 995–1011.