Supplementary sections to

Constructing Two-Level Designs by Concatenation of Strength-3 Orthogonal Arrays

Alan R. Vazquez[1], Peter Goos[1, 2], and Eric D. Schoen[1, 3]

[1]University of Antwerp, Belgium
[2]University of Leuven, Belgium
[3]TNO, Zeist, Netherlands

April 30, 2018

This document includes the following sections.

A Objective functions and fast update methods

The CC/VNS algorithm either optimizes the B_4 value or the F_4 vector of the concatenated design. The objective functions of the algorithm are derived from the B_4 value and the F_4 vector, but, for computational reasons, they are not the same. The objective function values must be evaluated after each change in the lower parent design. This section provides a detailed discussion of the way in which we evaluate the objective function at a low computational cost.

A.1 B_4 optimization

The direct calculation of B_4 in a k-factor design D with N runs and coded levels -1 and 1 requires the evaluation of all $k!/\left[4!(k-4)!\right]$ four-factor interaction contrast vectors. A computationally cheaper alternative was proposed by Butler (2003). He expressed the similarity of the runs in D with the matrix $T = DD^T$, and he showed that, for any orthogonal array, $M_4 = 24B_4 + k(3k-2)$, where $M_4 = N^{-2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}^4$ is the fourth moment of T. Clearly, minimizing M_4 is equivalent to minimizing B_4.

Let D_u and D_l be two two-level orthogonal arrays with n runs, m factors, and coded levels -1 and 1. Consider the concatenated design D of dimension $N \times m$, where $N = 2n$. We define the $n \times n$ matrices $A = D_u D_u^T$, $B = D_l\, D_l^T$, and $C = D_u D_l^T$. The similarity matrix T of the concatenated design D can then be partitioned as

$$T = \begin{bmatrix} A & C \\ C^T & B \end{bmatrix}.$$

Therefore, we can compute the fourth moment M_4 of design D as

$$M_4 = N^{-2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^4 + \sum_{i=1}^{n}\sum_{j=1}^{n} B_{ij}^4 + 2\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^4\right). \tag{1}$$

It is easy to show that the sum of all the elements A_{ij}^4 and B_{ij}^4 is invariant to sign-reversals of columns, column permutations, and row permutations in D_u and D_l. As a result, minimizing the M_4 value of the concatenated design D is equivalent to minimizing the sum of the elements C_{ij}^4 in (1). For this reason, the CC/VNS algorithm minimizes the following objective function to improve the B_4 value of the concatenated design:

$$b(D) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^4.$$

We further reduce the computations required as follows. We first note that the calculation of $b(D)$ implies a matrix multiplication of an $n \times m$ matrix D_u and an $m \times n$ matrix D_l^T. The number of computations required by this operation is $\left[2m-1\right] n^2$. So, every time the algorithm sign switches a column or swaps two columns in D_l, recalculating the matrix

$D_u D_l^T$ from scratch requires $[2m-1]\,n^2$ calculations to get the resulting objective value. A computationally cheaper approach is to change only the elements in C that correspond to the columns of D_l involved in a sign switch or swap.

Denote the columns of D_u and D_l as u_i and v_i, respectively, $i = 1, \ldots, m$. The matrix C can be expressed as a sum of matrices,

$$C = u_1\, v_1^T + \cdots + u_m v_m^T,$$

where $u_i\, v_i^T$ is a matrix of dimension $n \times n$. The matrix $u_i\, v_i^T$ is the contribution of the column u_i in D_u and the column v_i in D_l to C. This contribution is independent of the other columns in the parent designs. Thus, each time we sign switch or swap two columns in D_l, we just need to change their contributions and update the matrix C. The update formulas for C are as follows:

- Sign switch the column v_i: Update matrix C to $C' = C - 2\ u_i\ v_i^T$, where C' is the updated matrix. That is, subtract the contribution of the current column v_i and add the contribution of the new column $-v_i$. Note that the contribution of $-v_i$ is $-u_i\ v_i^T$. This procedure requires $2n^2 + n$ calculations: n multiplications to compute $-2\ u_i$, n^2 multiplications to compute $-2\ u_i\ v_i^T$, and n^2 summations to add the result to matrix C.

- Swap columns v_i and v_j: Update matrix C as $C' = C - (u_i\ v_i^T + u_j\ v_j^T) + (u_i\ v_j^T + u_j\ v_i^T)$. That is, remove the contribution of columns u_i and v_j in their current positions from C and add their new contribution due to their new positions in the lower design. Note that this procedure requires $8n^2 + 2n$ calculations: $2n$ multiplications to compute $-u_i$ and $-u_j$, $4n^2$ multiplications to compute $-u_i\ v_i^T, -u_j\ v_j^T, u_i\ v_j^T$ and $u_j\ v_i^T$, and $4n^2$ summations to add the results to matrix C.

The number of calculations required by the updating formulas for matrix C is clearly smaller than the matrix multiplication $D_u D_l^T$ when $m > 5$. More importantly, the number of calculations required by the updating formulas does not depend on the numbers of factors in the concatenated design. As a specific example, for two parent designs with 32 runs and 10 factors, the number of calculations required for a complete update of C when making a change in D_l is 19,456; the number of calculations required by the quick updating procedure is 2,080 for a sign switch of a column, and 8,256 for a swap between two columns. The difference between the number of calculations increases with the number of factors. Although the number of calculations saved by either updating formula suggests that both should be included in the implementation of our CC algorithm, the Matlab implementation only includes the updating formula for a sign switch. A computing time study (not shown) revealed that the updating formula for a swap of two columns requires the same or slightly more time than just changing the positions of the two columns and calculating matrix C from scratch. This is probably due to computer memory allocation.

A.2 F_4 optimization

Let the F_4 vector of a strength-3 design be $F_4 = (e_0, e_1, \ldots, e_r)$, where e_k denotes the frequency of the J_4-characteristics that equal $N - 16k > 0$ (Deng and Tang, 1999). Also, note that the run sizes of concatenated strength-3 designs are multiples of 16, so that $r = N/16 - 1$. We define the objective function, $f(D)$, as a linear combination of the elements of F_4, that is

$$f(D) = M_0 e_0 + \cdots + M_r e_r.$$

To mimic the G-aberration criterion, we ensure that $M_0 >> \cdots >> M_r$. More specifically, we use $M_i = 10^{5(r-i)}$, where $i = 0, 1, \ldots, r$.

In Matlab, the F_4 vector can be efficiently generated by using the two-factor interaction contrast matrix to calculate the J_4-characteristics. Let X and Y be the two-factor interaction contrast matrices for designs D_u and D_l, respectively. Without losing generality, let the columns of the matrix X (Y) be formed as the element-wise products $c_i \odot c_j$, where c_i is the i-th column of D_u (D_l), $i = 1, \ldots, m-1$ and $j = i+1, \ldots, m$. Then, the two-factor interaction contrast matrix of the concatenated design can be constructed as

$$Z = \left[X^T, Y^T\right]^T.$$

Now, consider the matrix

$$W = Z^T Z = X^T X + Y^T Y. \tag{2}$$

It is easy to show that each of the J_4-characteristics of the concatenated design D occurs in W six times. In Matlab, this procedure is far more efficient than computing the J_4-characteristics one by one using loop-based operations. Note that the CC/VNS algorithm performs changes to the lower design only. Therefore, we just need to compute matrix $Y^T Y$ and add it to the constant matrix $X^T X$. We can further improve the computing time by changing only the J_4-characteristics of columns involved in a change. We explain this below for a sign switch in a column of D_l.

Consider a column of D, $d_r = \left[u_r^T, v_r^T\right]^T$, where u_r and v_r are the rth columns of D_u and D_l, respectively. Let E and G be two submatrices of Z such that the columns of E include all interactions involving d_r and G contains the rest of the interactions. Then, the matrix $U = E^T G$ contains only the J_4-characteristics that involve column d_r and each of them appears three times. Note that we can express matrices E and G as

$$E = \left[E_u^T, E_l^T\right]^T \text{ and } G = \left[G_u^T, G_l^T\right]^T,$$

where E_u and G_u are submatrices of X and E_l and G_l are submatrices of Y. Then, we can write matrix U as

$$U = E_u^T G_u + E_l^T G_l. \tag{3}$$

From this expression, it is easy to see that the J_4-characteristics of the modified column $d_r' = \left[u_i^T, -v_i^T\right]^T$ can be obtained by multiplying matrix $E_l^T G_l$ in (3) by -1. For this reason, to compute the change in the F_4 vector of the concatenated design due to a sign switch of column v_r in D_l, we only need to remove the J_4-characteristics corresponding to column d_r, and add the J_4-characteristics corresponding to d_r'.

If the columns v_i and v_j of D_l are to be swapped, we update Y by computing the element-wise product of column $v_i \odot v_j$ with each of the columns in matrix Y that involve v_i or v_j.

B Algorithm performance evaluation

We implemented the CC/VNS algorithm in Matlab. In this section, we evaluate its performance for improving concatenated designs. We evaluate the impact of the two main components of our algorithm, the column change algorithm and the neighborhood structures of the VNS, on the B_4 value and the F_4 vector of the concatenated designs. We test our algorithm using five design cases involving three different run sizes and numbers of factors. Reported computing times relate to a standard CPU (Intel(R) Core(TM i7 processor, 2.8 GHz, 8 GB)).

B.1 Design cases

Table 1 shows the five design cases we used to evaluate the CC/VNS algorithm. The concatenated designs differ in run size, number of factors, and parent designs. All parent designs minimize the G_2-aberration criterion. The first instance, OA64One, requires the construction of a 64-run design with 16 factors by concatenating two different 16-factor 32-run parent designs that do not minimize the G-aberration criterion. The second instance, OA64Two, requires the construction of a 64-run design with 16 factors from two different 16-factor 32-run parent designs that both minimize the G-aberration criterion. The third instance, OA80, requires the construction of an 80-run concatenated design with 20 factors from different parent designs that both have minimum G-aberration. For the fourth instance, OA96One, we consider a 96-run concatenated design with 24 factors that is constructed by concatenating two 48-run OAs with different F_4 vectors. The last instance, OA96Two, is based on two identical minimum G-aberration 48-run OAs.

Table 2 shows the computing times required for 10 optimizations performed by the CC/VNS algorithm. For each of the cases in Table 1, Table 2 gives the averages and standard deviations of the computing times for the two objective functions minimized. Each optimization started with a random permutation of the lower design's columns and a random sign switch in these columns.

Clearly, it takes much more computing time to minimize the F_4 objective function than to minimize B_4. For the 96-run design cases with $m = 24$ factors, on average, more than one hour is needed for a single optimization. To construct designs that optimize the F_4

Table 1: Design cases used to evaluate the performance of the CC/VNS algorithm. The upper (D_u) and lower (D_l) parent designs have $N/2$ runs, m factors, a generalized resolution R and an F_4 vector as indicated. A dash as an element of the F_4 vector means that the corresponding J_4-characteristic does not exist. The concatenated designs have N runs and m factors. Labels of the parent designs come from the enumeration of Schoen et al. (2010).

			D_u				D_l			
Case	N	m	Label	R	$F_4(48,32,24,16,8)$	B_4	Label	R	$F_4(48,32,24,16,8)$	B_4
OA64One	64	16	2	4	$(-,76,0,256,0)$	140	3	4	$(-,44,0,384,0)$	140
OA64Two	64	16	4	4	$(-,28,0,448,0)$	140	5	4	$(-,28,0,448,0)$	140
OA80	80	20	2	4.4	$(-,0,285,0,4560)$	285	3	4.4	$(-,0,285,0,4560)$	285
OA96One	96	24	2	4	$(66,0,0,3960,0)$	506	60	4.67	$(0,0,0,4554,0)$	506
OA96Two	96	24	60	4.67	$(0,0,0,4554,0)$	506	60	4.67	$(0,0,0,4554,0)$	506

Table 2: Computing times for 10 optimizations performed by the CC/VNS algorithm. Average time ± standard deviation in seconds.

Instance	B_4	F_4
OA64One	4 ± 1.18	93.1 ± 17.2
OA64Two	3.6 ± 0.86	89.4 ± 23.4
OA80	19.1 ± 5.38	816.8 ± 268.8
OA96One	80.1 ± 14.98	4275 ± 1582.3
OA96Two	78.11 ± 11.95	3733 ± 1186.8

vector with run sizes $N > 96$ and $m > 24$, we have to restrict the size of the neighborhood N_4 to be $24!/\left[3!(24-3)!\right] = 2024$, the size of N_4 when $m = 24$, to keep the computing times for one iteration within 4 hours.

B.2 CC algorithm

The column change part of the CC/VNS algorithm is an algorithm in its own right. In this section, we demonstrate the effectiveness of the CC algorithm to minimize the B_4 value or the F_4 vector of the concatenated design. For each of the design cases listed in Table 1, we generate 1,000 random starting plans of the lower parent design and optimize the B_4 value or the F_4 vector of the concatenated designs with the CC algorithm only. We compare the results with 1,000 concatenated designs obtained from a random search. Each design is the overall best of 10,000 randomly generated, concatenated designs.

Figure 1 presents box plots for the B_4 values of the concatenated designs found by either strategy. There are separate panels in the figure for each of the design cases. We display the medians as dots in all box plots in this document. Figure 1a is concerned with OA64One. Here, the medians of the B_4 values for the random search and the CC algorithm both equal 65.5. For the CC algorithm, the median coincides with the upper quartile so that few of the B_4 values produced by that algorithm exceed the median. For the random search, the median coincides with the lower quartile, so that most of the B_4 values produced by the random search will be larger than the median. This shows that we are better off by using the CC algorithm than by conducting a random search.

The case of OA64Two is illustrated in Figure 1b. Here, the median B_4 value provided by the CC algorithm is smaller than the median B_4 value of the random search. Finally, for the larger cases, Figures 1c−1e clearly show that the majority of the B_4 values obtained by the CC algorithm are smaller than those obtained by the random search. We conclude that the CC algorithm outperforms a random search, and that the improvement over the random search increases with the size of the design.

To evaluate the F_4 optimization, we check the generalized resolution as well as the $f(D)$ values of the concatenated designs. Figure 2 shows the distribution of the generalized resolutions of the concatenated designs from the random search and the CC algorithm. Figure 2a shows that, for the OA64One case, the CC algorithm produces substantially more designs with a generalized resolution of 4.5 than the random search. The fact that this resolution is reached in only 10% of the runs of the algorithm suggests that at least 10 restarts of the CC algorithm are needed for an optimal result with the stand alone CC algorithm. Regarding the OA64two case, Figure 2b shows that the CC algorithm and the random search resulted in the same number of concatenated designs with generalized resolution 4.25 and with generalized resolution 4.5. For the 80-run case, Figure 2c shows that the CC algorithm only produced concatenated designs with a generalized resolution of 4.6, while the random search generated 85 designs with a generalized resolution of 4.4. For the OA96One case, the CC algorithm created 105 concatenated designs with a generalized resolution of 4.5, whereas the random search produced only designs with a generalized resolution of 4.33; see Figure 2d. Finally, all concatenated designs for the OA96Two case had a resolution of 4.67, regardless of whether the CC algorithm or the random search was used. For this reason, Figure 2 does not include a separate panel for the OA96Two case.

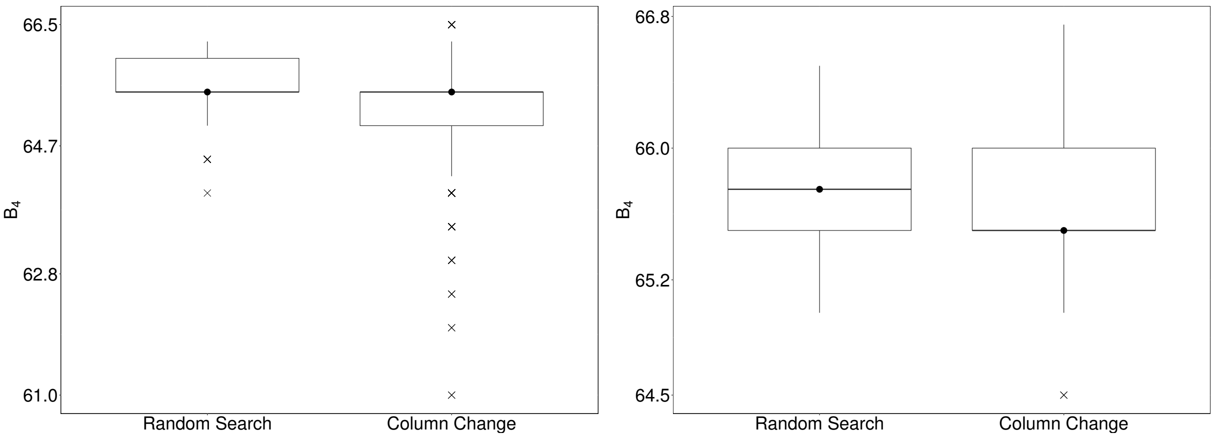
We now turn to the $f(D)$ value of the designs that optimize the F_4 vector. Recall from Section A.2 that $f(D)$ is a linear combination of the elements of the F_4 vector, in which the frequencies of large J_4-characteristics receive a larger weight than those of small J_4-characteristics. Low values for the $f(D)$ objective function thus imply that the design has a high generalized resolution and that the frequency of the largest J_4-characteristic is small. So the $f(D)$ value is able to distinguish designs that have the same generalized resolution.

Figure 3 shows the $f(D)$ values for the concatenated designs from the random search and the CC algorithm. Figure 3b and Figure 3c include only designs with a generalized resolution of 4.5 and 4.6, respectively, because, otherwise, the weights for the large J_4-characteristics would distort the figure. Figure 3a shows that the medians of the concatenated designs from both approaches are the same. For the CC algorithm, the median coincides with the upper quartile, so that few of the $f(D)$ values exceed the median. For the random search, the median coincides with the lower quartile, so most of the $f(D)$ values are larger than the median. So, we are better off by using the CC algorithm than by conducting a random search. Figures 3b–3e for the other design cases show that the CC algorithm generally generated concatenated designs with smaller $f(D)$ values and thus better F_4 vectors than the random search.

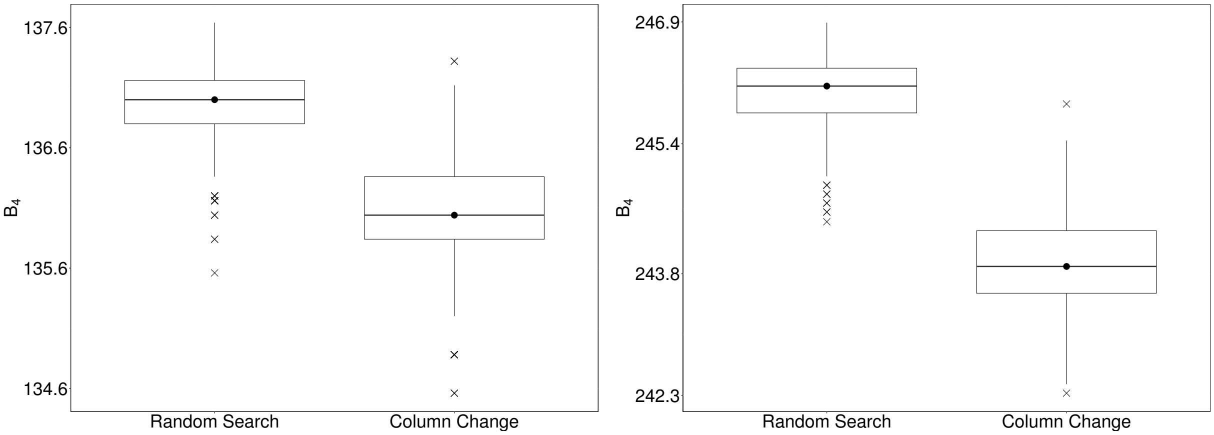
B.3 Neighborhood structures

In this section, we investigate how important each added neighborhood is for the performance of the CC/VNS algorithm. We generated concatenated designs with versions of the CC/VNS algorithm including only neighborhood N_1, including neighborhood N_1 and N_2, and so on. For the 64-run cases, the 80-run case, and the 96-run cases, we generated 500 concatenated designs. For the 96-run cases in which the F_4 vector was optimized, we only generated 100 concatenated designs.

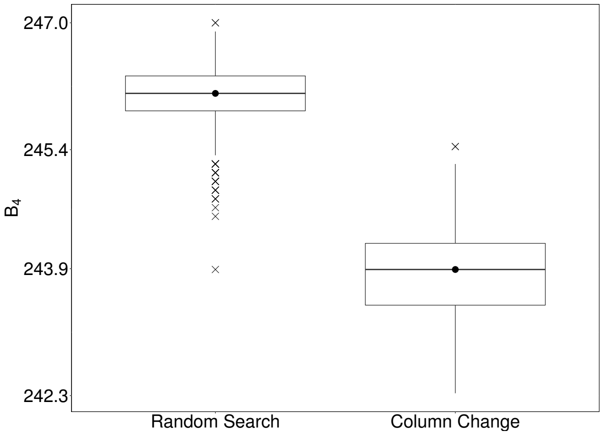
Figure 4 shows box plots of the B_4 values of the concatenated designs when adding one neighborhood at a time. For the OA64One case, Figure 4a shows a decrease in the



(a) OA64One (b) OA64Two



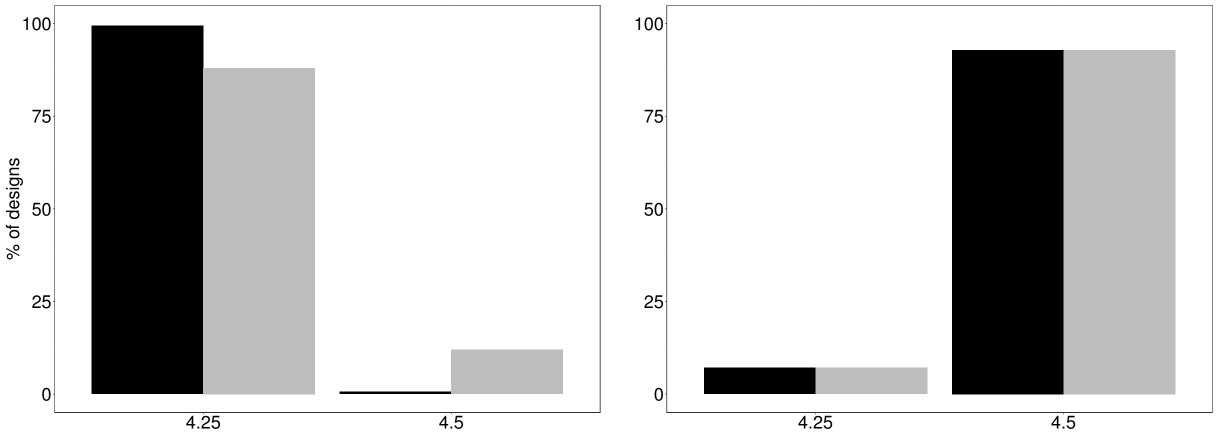
(c) OA80 (d) OA96One



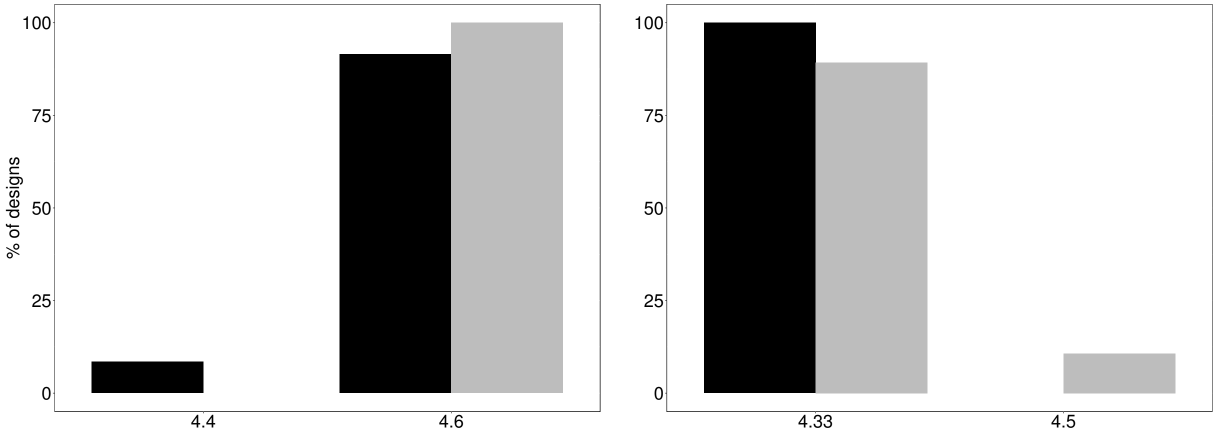
(e) OA96Two

Figure 1: Performance of the column change algorithm and a random search strategy in terms of minimizing the B_4 value. Each boxplot involves 1,000 optimized designs.

median and the variance of the B_4 values when neighborhoods two and three are introduced successively. Using neighborhoods $N_1 - N_3$, almost all concatenated designs have a B_4 value



(a) OA64One (b) OA64Two

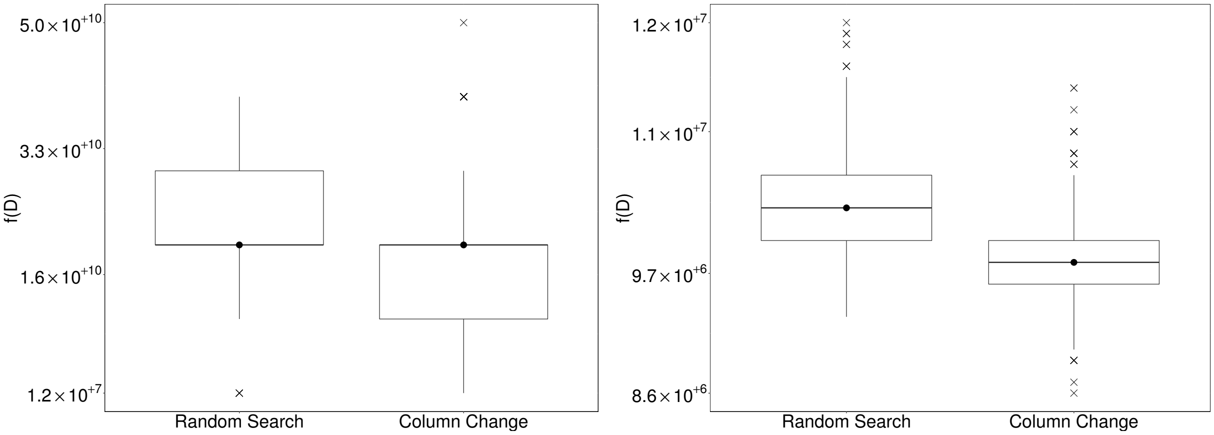


(c) OA80 (d) OA96One

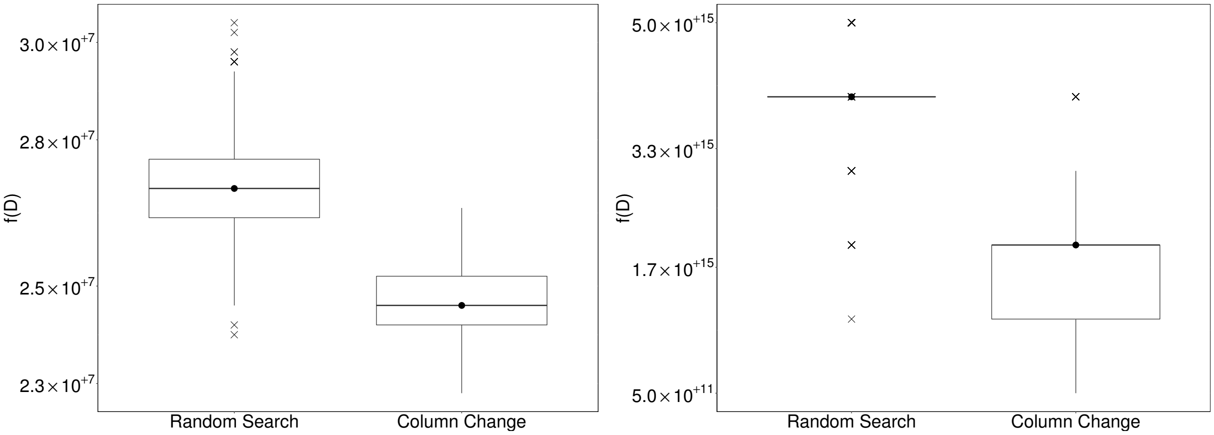
Figure 2: Generalized resolution for concatenated designs resulting from random search (black) and the column change algorithm (gray) for four of the design cases in Table 1. 100% corresponds to 1,000 optimized designs.

of 61. When we also include the fourth neighborhood, half of the resulting concatenated designs for this case have a B_4 value smaller than 61. For the OA64Two case, Figure 4b shows that introducing the second neighborhood does not lead to smaller B_4 values in the concatenated design. However, successively including neighborhoods three and four leads to large numbers of concatenated designs with B_4 values lower than 65. Figure 4c shows a decrease in the B_4 values of the concatenated designs for case OA80 when the second and the fourth neighborhood are added; there seems to be no effect of the third neighborhood in this case. Figures 4d and 4e clearly show a shift in the median and the distribution of the B_4 values produced by each additional neighborhood for the 96-run cases.

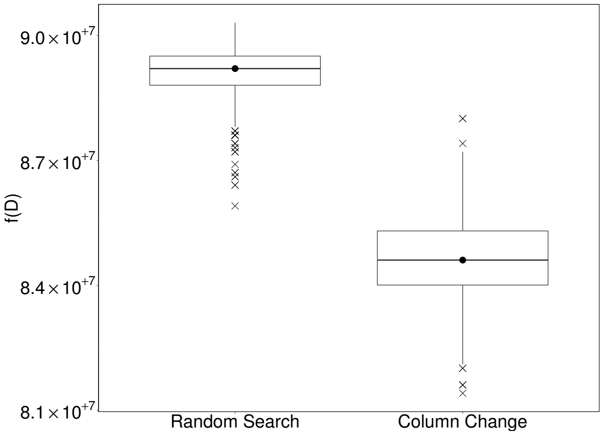
The effect of the successive inclusion of the four neighborhoods on the $f(D)$ value is



(a) OA64One (b) OA64Two



(c) OA80 (d) OA96One



(e) OA96Two

Figure 3: Performance of the column change algorithm and a random search strategy in terms of the $f(D)$ value. Figures (a), (d) and (e) show $f(D)$ values for all designs resulting from 1,000 starts. In Figures (b) and (c), designs with a generalized resolution of 4.25 (OA64Two) or 4.4 (OA80) are disregarded.

shown in Figure 5. For clarity of presentation, we removed the OA64One and OA96One designs with generalized resolutions of 4.25 and 4.5, respectively. So, all concatenated designs involving 64, 80, and 96 runs shown in the figure have generalized resolutions of 4.5, 4.6 and 4.67, respectively. The figure shows that a successive inclusion of the neighborhoods $N_1 - N_4$ generally improves the objective function value. The improvement due to the third neighborhood, however, is substantial only in the OA96One case. For that case, the median $f(D)$ value over 500 designs decreases from 5×10^{11} to 4.8×10^{11}. As neighborhood N_3 is beneficial in at least one case, we retain this neighborhood in the CC/VNS algorithm.

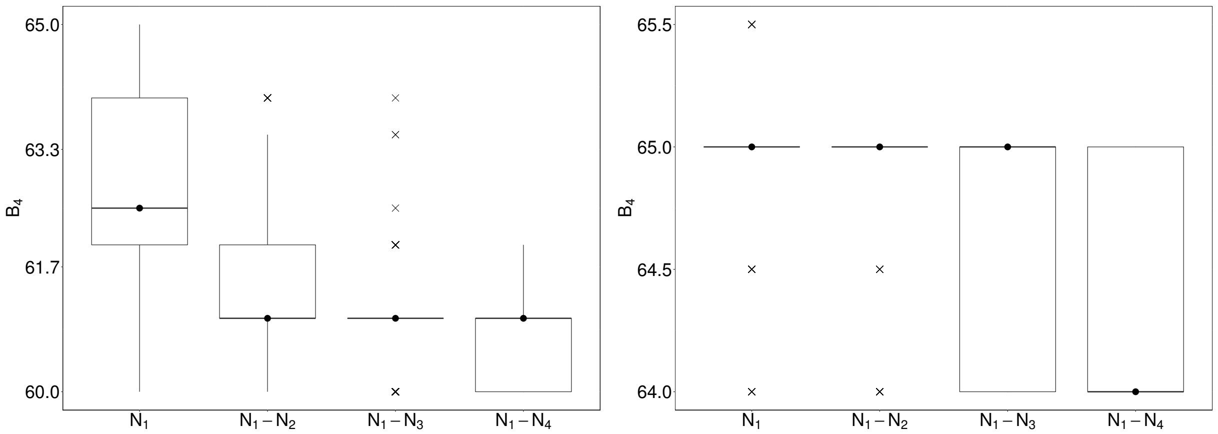
B.4 Performance for 128-run designs

We further test the potential of the CC/VNS algorithm by constructing 128-run designs with 10, 15, 20, 25 and 30 factors from 64-run parents with the same number of factors. We tested B_4 optimization as well as F_4 optimization.

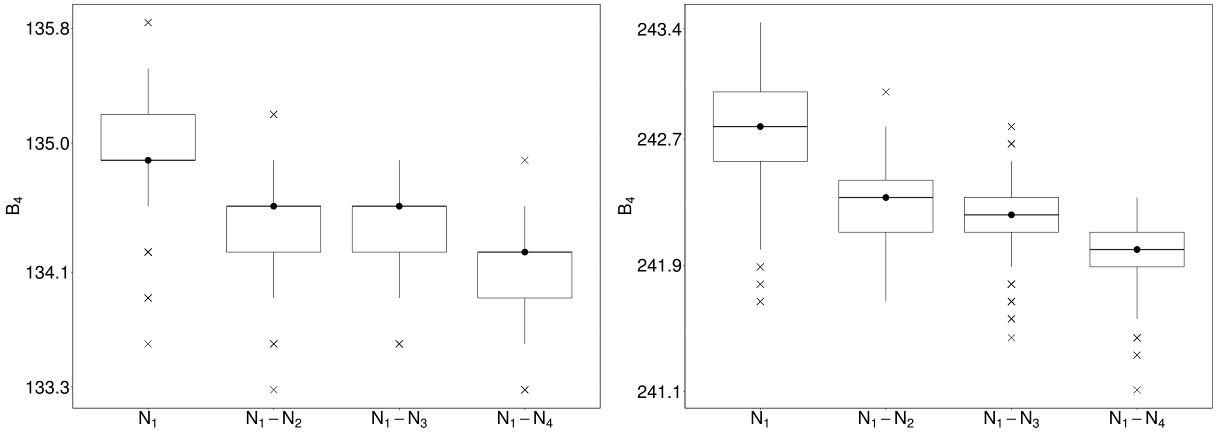
We obtained suitable parent designs from three different sources. The first one is the complete collection of regular 64-run designs of strength 3 (Chen et al., 1993). We used the minimum aberration (MA) designs as parent designs and we label these designs 64.m.MA, where m is the number of factors. Because all the non-zero J_4-characteristics equal 64 in these designs, we use them for B_4 optimization only, as they are less likely to result in 128-run designs with minimal F_4 vectors.

The second source of parent designs is the collection of nonregular designs based on quaternary linear codes (QLC) found by Xu and Wong (2007). That collection includes one or two 64-run designs for each number of factors up to 56. Whenever two designs are given, one design has the best B_4 value and the other has the best F_4 vector of the two. The parent designs are labeled 64.m.QLC when there is a single QLC design, or 64.m.QLC/B4 and 64.m.QLC/F4 in case there are two different designs.

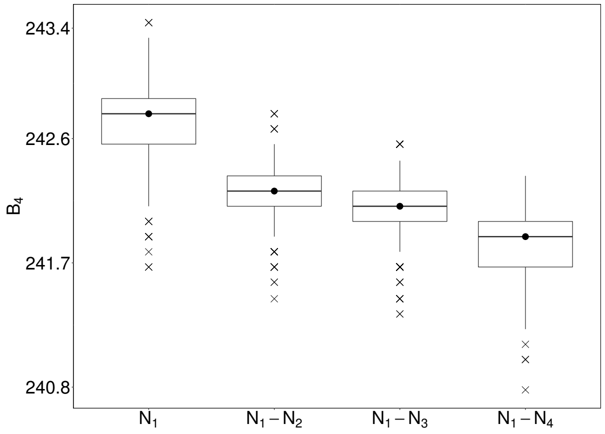
Finally, we used projections of the 64-run strength-3 OA with 32 factors constructed by folding-over the Paley Hadamard matrix of order 32 (Sloane, 1999). To find designs with 25 and 30 factors, we evaluated all projections, while for 10, 15 and 20 factors, we evaluated 50,000 random projections. The projections with the best F_4 vectors were used as parent



(a) OA64One (b) OA64Two



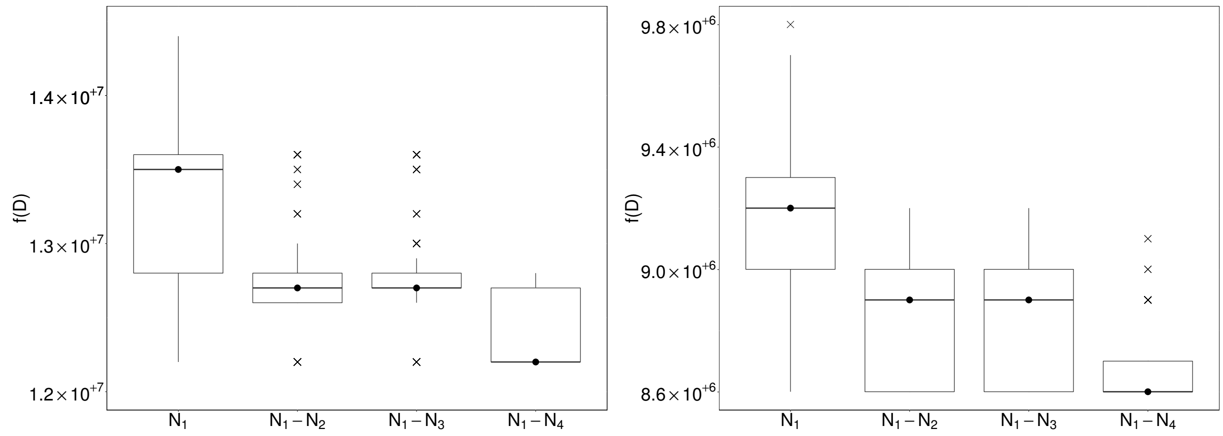
(c) OA80 (d) OA96One



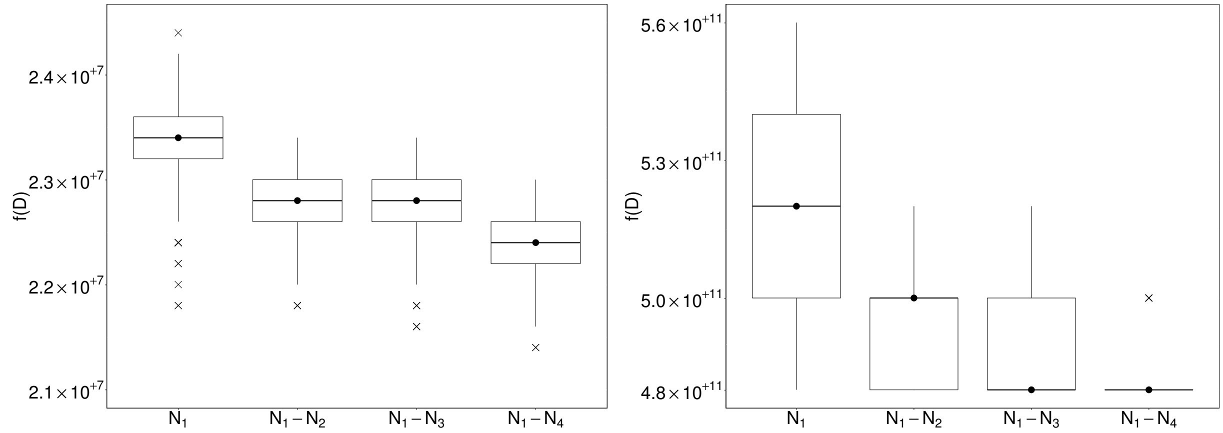
(e) OA96Two

Figure 4: B_4 values for 500 concatenated designs produced by the CC/VNS algorithm using neighborhoods N_1, $N_1 - N_2$, $N_1 - N_3$, or $N_1 - N_4$.

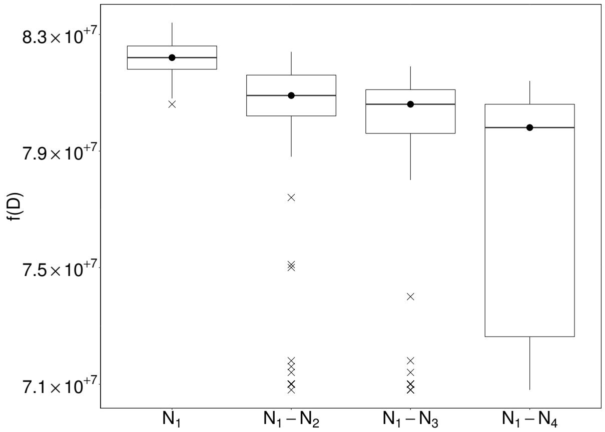
designs. Details are shown in Section C. These parent designs are labeled 64.m.P. We use these designs for F_4 optimization only as they have larger B_4 values than the other parent



(a) OA64One (b) OA64Two



(c) OA80 (d) OA96One



(e) OA96Two

Figure 5: $f(D)$ values for concatenated designs produced by the CC/VNS algorithm using neighborhoods N_1, $N_1 - N_2$, $N_1 - N_3$, or $N_1 - N_4$. The box plots in values (a), (b), (c), (d) and (e) show results for 470, 500, 500, 93 and 100 designs, respectively.

designs.

Table 3: Performance of the CC/VNS algorithm for constructing 128-run designs that optimize the B_4 value.

Parent design	B_4 parent	B_4 concatenation	Percentage of iterations	Percentage of all plans
64.10.MA	2	0	100	0.089
64.10.QLC	2	0–0.5	88	0.082
64.15.MA	30	12	100	0.000
64.15.QLC	33	12 –13.375	65	0.000
64.20.MA	125	52	100	0.000
64.20.QLC/B4	125	52	100	0.000
64.25.MA	435	198–205	45	0.000
64.25.QLC/B4	435	198–205.5	6	0.000
64.30.MA	945	447–450.5	59	0.000
64.30.QLC	945	447–455.4	13	0.000

Table 3 shows the results for 100 iterations of the CC/VNS procedure when the objective is to optimize the B_4 value. The second column shows the B_4 values of parent designs. The third column shows the range of the B_4 values obtained over the 100 iterations. The fourth column shows the percentage of the iterations in which the design with the smallest B_4 value was found. For the cases with up to 20 factors, 10 iterations should suffice to find the best design at least once. The cases with 25 and 30 factors are clearly more demanding. However, the range of B_4 values is quite narrow. It seems therefore reasonable to use 10 iterations for these cases as well. The last column shows that only a very small proportion of all possible plans is visited by the CC/VNS algorithm to find the final concatenated design.

Table 4 shows the results for 100 iterations of the CC/VNS procedure when the objective is to optimize the F_4 vector. The second and third columns show the generalized resolution (GR) of the parent and concatenated designs, respectively. The 10-factor parent designs have resolutions of 4.75 and 4.5. For both cases, the best concatenated 10-factor designs have GRs of 5. In addition, QLC parent designs with 15, 20, 25 and 30 factors have a GR of 4, while the corresponding concatenated designs have an improved GR of 4.5. For all but one case, the parent designs obtained from projections of the folded-over 32-run

Table 4: Performance of the CC/VNS algorithm for constructing 128-run designs under F_4 optimization.

Parent design	GR		$F_4^{\max}$		Percentage of iterations	Percentage of all plans
	parent	concatenation	parent	concatenation		
64.10.P	4.75	5	96	0	53	0.173
64.10.QLC	4.5	5	8	0	90	0.077
64.15.P	4.75	4.75	726	131-139	1	0.000
64.15.QLC	4	4.5	21	32-34	25	0.000
64.20.P	4.75	4.75	2640	575-602	1	0.000
64.20.QLC/F4	4	4.5	94	160-170	1	0.000
64.25.P	4.75	4.75	6974	1682-1710	1	0.000
64.25.QLC/F4	4	4.5	247	460-483	4	0.000
64.30.P	4.75	4.75	15120	1099-1153	1	0.000
64.30.QLC	4	4.5	561	3800-3835	3	0.000

Paley matrix have the same GR as the corresponding concatenated designs. The next two columns in Table 4 show the frequency of the largest J_4-characteristic of the parent design and the concatenated design, respectively. The 100 concatenated designs for each case with 15 or more factors show a range of frequencies for the maximum J_4-characteristic. The best value typically occurs only a few times. However, the range of the frequencies is rather narrow, so that any concatenated designs produced by the CC/VNS algorithm is actually a good design. The last column of Table 4 shows again that only a very small proportion of all possible plans is visited when constructing the concatenated design.

C Parent designs

We obtained the 32-run parent designs for the 64-run concatenated designs from the complete catalogs available in Schoen et al. (2010). The parent designs we considered for the 64-run designs that optimize the B_4 value were the 32-run OAs that have a minimum B_4 value. The parent designs for 64-run designs that optimize the F_4 vector were chosen from the top three 32-run OAs in terms of the F_4 vector. That is, we sorted the F_4 vectors of all 32-run OAs and then selected the first three designs. For 12 and 13 factors, there are two OAs that can have the last position in the top three. We selected one of these two designs at random.

The 64-run designs used to construct 128-run designs that optimize the B_4 value include the 64-run regular minimum aberration designs (Chen et al., 1993), the 64-run designs constructed from quaternary linear codes (Xu and Wong, 2007), and our best 64-run concatenated designs in terms of the B_4 value. We label these designs 'ma.l', 'xw.l', and 'coa.l', respectively, where l is the label used by the aforementioned authors.

For 128-run designs that optimize the F_4 vector, we used our own 64-run concatenated designs and projections of the 64-run strength-3 OA with 32 factors constructed by folding-over the Paley Hadamard matrix of order 32 (Sloane, 1999). For $m < 10$ and $m > 22$, we evaluated all projections, while for $10 \leq m \leq 22$, we evaluated 50,000 random projections. The projections with the best F_4 vectors were used as parent designs. Table 5 shows the best F_4 vectors for $9 \leq m \leq 32$ and the m columns from the folded-over Paley matrix required to obtain these vectors. The number of degrees of freedom for two-factor interaction equals 30 for $m = 11$ and 31 for all other designs. For $9-11$ factors, some of our best 64-run concatenated designs outperform the designs constructed from projections of the folded-over Paley Hadamard matrix of order 32 in terms of the G-aberration criterion. For this reason, for 9 and 10 factors, we considered our best 64-run designs in terms of the B_4 value and, for 11 factors, our best 64-run design in terms of the F_4 vector, as parent designs.

D Tables of concatenated designs

We present concatenated designs with 64 and 128 runs in Tables 6, 7 and 8, respectively. The designs are labeled as $k.b$ or $k.f$, where $k = m + 1$ is the number of factors in the concatenated design, b corresponds to designs that minimize the B_4 value and f to designs that sequentially minimize the F_4 vector. All concatenated designs shown include the indicator factor $\mathbf{z} = \left[\mathbf{1}_{N/2}, -\mathbf{1}_{N/2}\right]^T$, where $\mathbf{1}_{N/2}$ is a $N/2 \times 1$ column vector of ones and N is the run size of the concatenated design. This factor increases the number of degrees of freedom for two-factor interactions by $m - 1$; all interactions involving this factor are clear. There are separate tables for 128-run designs that optimize the B_4 value and for designs that optimize the F_4 vector. The tables report the generalized resolution (R), the F_4 vector, the B_4 value, the degrees of freedom for two-factor interactions (df), and the

Table 5: Projections from the folded-over Paley Hadamard matrix of order 32 used to construct F_4-optimized 128-run designs. $F_4(64, 48, 32) = (0, 0, 0)$ for all designs.

m	Label	$F_4(16)$	Columns
9	P9	58	1 2 3 4 5 11 12 19 30
10	P10	96	3 9 11 13 19 21 23 28 29 31
11	P11	160	2 3 6 8 10 11 14 16 24 25 30
12	P12	252	7 8 9 10 14 15 17 20 21 24 26 28
13	P13	370	4 5 6 9 10 13 21 24 25 28 29 30 31
14	P14	526	1 4 6 7 8 12 17 20 22 24 25 28 29 30
15	P15	726	1 2 5 7 11 13 14 15 16 21 23 24 25 28 30
16	P16	978	2 3 4 7 9 11 13 15 19 21 22 24 27 28 29 30
17	P17	1286	5 7 8 9 13 15 16 17 19 20 21 22 23 24 26 27 29
18	P18	1666	2 5 7 8 10 11 14 15 16 17 18 19 20 21 22 24 28 31
19	P19	2112	1 4 6 7 8 10 11 13 14 15 17 18 20 23 24 27 29 30 31
20	P20	2640	1 2 3 4 7 8 9 11 12 13 15 17 23 24 25 26 27 28 30 31
21	P21	3280	1 2 3 4 5 8 10 11 12 14 15 17 19 20 22 23 24 25 28 29 30
22	P22	4018	1 4 5 7 9 10 12 15 16 17 18 19 20 21 22 23 24 25 26 27 30 31
23	P23	4874	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 19 20 22 24 29 30
24	P24	5854	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 23 25 30
25	P25	6974	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 25 27
26	P26	8244	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 25 29 30
27	P27	9680	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 25 27 28 30
28	P28	11296	$1-28$
29	P29	13104	$1-29$
30	P30	15120	$1-30$
31	P31	17360	$1-31$
32	P32	19840	$1-32$

upper (D_u) and lower parents (D_l) of the concatenated designs. The columns γ and δ in the tables denote the columns in D_l of which the signs have to be switched, and the column permutation required to obtain the final design after the sign switch, respectively.

The indicator factor can be used as a blocking factor, in which case the number of degrees of freedom for interactions should be decreased by $m - 1$. If the concatenated design is made up from different parent designs, we recommend to run first the parent design with the smallest B_4 value or the best F_4 vector, or the one with the largest number of degrees of freedom for interactions, depending on the interest of the experimenter. There are four cases in which our 64-run designs are made from different parent designs and five cases in which our 128-run designs are constructed from different parents.

For 64 runs, the concatenated designs based on different parent designs are 9.b, 10.b, 12.f and 16.b. Table 6 highlights the parent designs with the best B_4 value (^{a}), the best F_4 vector (^{b}), and the largest number of degrees of freedom for two-factor interactions (^{c}). For 9 and 10 factors, both parent designs have the same B_4 value and number of degrees of freedom for interactions but one is best in terms of the G-aberration criterion. For 12 and 16 factors, the parent designs have the same number of degrees of freedom for interactions and the same B_4 value, but parent 20 (11 factors) and 3 (15 factors) have a better F_4 vector than parents 21 and 2, respectively.

For 128 runs, the concatenated designs based on different parent designs are 25.b, 27.b, 28.b, 30.b, and 32.b. Each of these designs consists of a QLC design and a MA design. In each, both parent designs have the same B_4 value and the same number of degrees of freedom for two-factor interactions, but the QLC designs have a smaller G-aberration. If the indicator factor is used as a blocking factor, we therefore recommend starting with the QLC design.

The steps required to construct the concatenated designs from Tables 6 and 8 are:

1. Obtain the upper (D_u) and lower (D_l) parent designs in m factors.
2. Switch the signs of the columns γ in D_l. Denote the resulting design by D_s.
3. Arrange the columns of D_s in the order indicated by δ. Denote the resulting design by D_{sp}.
4. Concatenate D_u and D_{sp} to create design D.
5. Append the indicator factor column $\mathbf{z}$ to D to obtain the final design involving $k = m + 1$ factors.

Example. To construct the 64-run design for 10 factors that optimizes the B_4 value, we take OA$(32, 2^9, 3)$ with ID 34 from Schoen et al. (2010) as the lower parent design. Next, we generate design D_s by reversing the signs of columns 3, 5, 6, 7, and 8 in that design. Subsequently, we generate design D_{sp} by arranging the columns of design D_s in the following order: 6, 3, 4, 5, 2, 8, 9, 1, 7. That is, column 6 of D_s is the first column of design D_{sp}, column 3 of D_s is the second column of D_{sp}, column 4 of D_s is the third column of D_{sp}, and so on. Next, we concatenate the 9-factor orthogonal array ID 27 from Schoen et al. (2010)

with this design. Finally, we add the extra factor $\mathbf{z} = [\mathbf{1}_{32}, -\mathbf{1}_{32}]^T$ to the concatenated design to produce the 64-run design for 10 factors that optimizes the B_4 value. This design has a generalized resolution of 4.75, $F_4(64, 48, 32, 16) = (0, 0, 0, 32)$, $B_4 = 2$, and 45 degrees of freedom for two-factor interactions. So, the design permits estimation of all two-factor interactions along with the main effects.

Table 6: Concatenated designs with 64 runs. ^{a}: parent design with best B_4 value; ^{b}: parent with the best F_4 vector; ^{c}: parent with the largest number of degrees of freedom for two-factor interactions.

Design	D_u	D_l	R	$F_4(64, 48, 32, 16)$	B_4	df	γ	δ
9.b	23^{ac}	32^{abc}	4.75	(0, 0, 0, 16)	1	36	$1, 5, 8$	$4, 1, 8, 3, 6, 2, 5, 7$
9.f	32	32	4.75	(0, 0, 0, 16)	1	36	$2, 3, 4, 5$	$1, 4, 2, 8, 5, 3, 7, 6$
10.b	27^{ac}	34^{abc}	4.75	(0, 0, 0, 32)	2	45	$3, 5, 6, 7, 8$	$6, 3, 4, 5, 2, 8, 9, 1, 7$
10.f	34	34	4.75	(0, 0, 0, 32)	2	44	$1, 5, 8, 9$	$6, 4, 8, 3, 2, 1, 7, 9, 5$
11.b	20	20	4.5	(0, 0, 16, 0)	4	48	$1, 2, 3, 5, 6, 8, 9, 10$	$6, 5, 7, 9, 4, 2, 10, 8, 1, 3$
11.f	32	32	4.75	(0, 0, 0, 108)	6.75	40	$1, 2, 3, 6, 7, 8, 10$	$1, 4, 3, 2, 7, 10, 6, 8, 5, 9$
12.b	10	10	4.5	(0, 0, 21, 72)	9.75	41	$1, 2, 3, 4, 5, 8, 9$	$3, 7, 6, 4, 2, 1, 5, 10, 11, 8, 9$
12.f	20^{abc}	21^{ac}	4.5	(0, 0, 5, 154)	10.88	41	$1, 2, 3, 5, 6, 9, 11$	$10, 1, 7, 2, 5, 11, 4, 9, 8, 6, 3$
13.b	8	8	4.5	(0, 0, 36, 96)	15	42	$2, 4, 7, 8, 9, 10, 11$	$7, 6, 4, 8, 3, 5, 1, 2, 11, 12, 10, 9$
13.f	21	21	4.5	(0, 0, 10, 216)	16	42	$1, 4, 5, 6$	$12, 7, 10, 2, 1, 9, 3, 4, 11, 8, 5, 6$
14.b	2	2	4.5	(0, 0, 88, 0)	22	43	$1, 2, 4, 5, 9, 11, 12, 13$	$7, 5, 6, 8, 11, 12, 9, 10, 2, 3, 1, 4, 13$
14.f	12	12	4.5	(0, 0, 24, 292)	24.25	43	$1, 6, 10, 12, 13$	$13, 4, 3, 1, 8, 5, 6, 10, 7, 12, 11, 9, 2$
15.b	2	2	4.25	(0, 8, 68, 184)	33	44	$1, 2, 3, 4, 5, 6, 9, 12, 13$	$14, 10, 13, 11, 3, 8, 4, 5, 6, 7, 1, 2, 9, 12$
15.f	8	8	4.5	(0, 0, 38, 406)	34.88	44	$3, 7, 10, 12, 13, 14$	$11, 9, 6, 14, 2, 5, 13, 7, 1, 10, 3, 4, 12, 8$
16.b	2^{ac}	3^{abc}	4	(9, 0, 72, 288)	45	45	$1, 3, 5, 8, 9, 10, 11, 12, 13, 14$	$8, 7, 1, 2, 12, 11, 10, 9, 4, 3, 5, 6, 13, 14, 15$
16.f	5	5	4.5	(0, 0, 57, 552)	48.75	45	$1, 3, 4, 8, 11, 12, 13$	$5, 2, 12, 14, 9, 8, 11, 13, 10, 7, 1, 6, 4, 3, 15$
17.b	3	3	4	(12, 0, 96, 384)	60	46	$1, 2, 3, 4, 7, 8, 10, 11, 14, 16$	$15, 16, 14, 13, 9, 10, 12, 11, 4, 3, 6, 5, 7, 8, 1, 2$
17.f	4	4	4.5	(0, 0, 83, 708)	65	46	$1, 8, 9, 10, 11$	$11, 13, 12, 1, 8, 7, 9, 16, 10, 14, 5, 15, 3, 2, 4, 6$

E 128-run designs of strength 4

It is known that strength-4 128-run designs exist with up to 15 factors; see Hedayat et al. (1999) for the construction of the 15-factor design. These designs necessarily consist of two concatenated strength-3 64-run designs to which an extra factor is appended. Therefore, provided the right 64-run parent designs are used as input, the CC/VNS algorithm should be able to construct strength-4 128-run designs. The parent designs we used for concatenating 128-run designs that optimize the B_4 value are the regular minimum aberration designs (Chen et al., 1993), the designs constructed from quaternary linear codes (Xu and Wong, 2007), and our own 64-run designs that minimize the B_4 value. For the 128-run designs that optimize the F_4 vector, we used the best projections of the folded-over 32-run

Table 7: 128-run concatenated designs that optimize the F_4 vector. $F_4(128, 112, 96, 80, 64, 48) = (0, 0, 0, 0, 0)$ for all designs.

Design	D_u	D_l	R	$F_5(96, 64, 32)$	B_5	df	γ	δ
10.f	coa.9.b	coa.9.b	5.25	(1, 4, 23)	3	45	6, 9	1, 9, 8, 4, 5, 2, 3, 7, 6
11.f	coa.10.b	coa.10.b	5.25	(2, 8, 46)	6	55	1, 4, 9	1, 7, 8, 9, 6, 5, 2, 3, 4, 10
Design	D_u	D_l	R	$F_4(32, 16)$	B_4	df	γ	δ
12.f	coa.11.f	coa.11.f	4.75	(1, 150)	2.41	66	1, 2, 7, 9, 10	10, 11, 6, 8, 9, 4, 7, 3, 2, 1, 5
13.f	P12	P12	4.75	(32, 286)	6.47	74	1, 3, 5	10, 11, 7, 3, 8, 2, 4, 9, 12, 5, 1, 6
14.f	P13	P13	4.75	(52, 408)	9.63	75	2, 8, 9, 12, 13	11, 8, 6, 1, 9, 4, 7, 2, 5, 12, 3, 10, 13
15.f	P14	P14	4.75	(88, 550)	14.09	76	1, 3, 5, 8, 11, 12, 13	6, 13, 8, 3, 5, 2, 10, 4, 14, 7, 1, 11, 9, 12
16.f	P15	P15	4.75	(131, 752)	19.94	77	2, 3, 4, 7, 8, 10, 11, 12, 13, 14	2, 5, 13, 8, 14, 10, 4, 6, 12, 9, 1, 7, 15, 3, 11
17.f	P16	P16	4.75	(189, 1018)	27.72	78	2, 3, 6, 7, 8, 11, 12, 13, 14	10, 3, 11, 8, 5, 7, 13, 16, 1, 6, 2, 9, 14, 15, 12, 4
18.f	P17	P17	4.75	(263, 1284)	36.5	79	3, 6, 8, 9, 10, 11, 13, 15, 16, 17	5, 11, 3, 14, 7, 9, 1, 12, 2, 17, 13, 8, 6, 10, 15, 16, 4
19.f	P18	P18	4.75	(355, 1672)	48.31	80	1, 7, 8, 10, 11, 12, 18	15, 14, 4, 11, 1, 7, 17, 13, 5, 6, 12, 18, 9, 2, 16, 8, 3, 10
20.f	P19	P19	4.75	(457, 2106)	61.47	81	1, 2, 6, 7, 10, 11, 13, 14, 16, 17, 18, 19	17, 12, 16, 19, 4, 3, 9, 13, 14, 11, 1, 7, 15, 5, 10, 8, 2, 18, 6
21.f	P20	P20	4.75	(584, 2656)	78	82	1, 2, 4, 6, 7, 8, 10, 12, 16, 17	5, 7, 18, 10, 2, 11, 3, 19, 12, 16, 9, 6, 4, 20, 17, 14, 1, 15, 8, 13
22.f	P21	P21	4.75	(753, 3188)	96.88	83	2, 3, 4, 7, 8, 9, 10, 13, 15, 16, 19, 21	1, 4, 15, 21, 16, 8, 17, 7, 20, 10, 11, 6, 14, 2, 5, 18, 9, 3, 19, 13, 12
23.f	P22	P22	4.75	(942, 3868)	119.31	84	2, 5, 6, 7, 8, 11, 12, 14, 17, 18, 20, 21, 22	5, 18, 22, 14, 13, 9, 15, 4, 8, 20, 19, 7, 2, 3, 12, 6, 11, 16, 10, 1, 17, 21
24.f	P23	P23	4.75	(1150, 4708)	145.44	85	3, 6, 7, 10, 11, 15, 17, 20	11, 16, 9, 8, 10, 18, 6, 5, 7, 14, 12, 19, 3, 13, 15, 23, 1, 17, 20, 22, 21, 4, 2
25.f	P24	P24	4.75	(1405, 5592)	175.19	86	1, 2, 3, 5, 8, 11, 12, 16, 17, 19, 21, 22, 23, 24	15, 20, 5, 21, 8, 22, 19, 12, 7, 18, 23, 10, 14, 9, 24, 1, 6, 16, 3, 11, 4, 17, 13, 2
26.f	P25	P25	4.75	(1695, 6620)	209.38	87	2, 3, 7, 9, 10, 13, 14, 16, 17, 19, 20, 21, 23, 25	7, 18, 24, 25, 15, 8, 2, 20, 22, 19, 14, 13, 1, 6, 11, 17, 12, 21, 16, 9, 3, 5, 23, 4, 10
27.f	P26	P26	4.75	(2018, 7802)	248.03	88	2, 3, 8, 9, 10, 11, 13, 15, 21, 22, 26	24, 2, 10, 25, 14, 18, 19, 9, 4, 8, 20, 16, 11, 15, 17, 3, 1, 7, 21, 23, 22, 5, 13, 26, 6, 12
28.f	P27	P27	4.75	(2386, 9228)	293.31	89	6, 7, 8, 10, 11, 12, 14, 15, 19, 20, 21, 22, 24, 26	26, 3, 27, 5, 6, 17, 1, 21, 18, 24, 13, 4, 14, 23, 25, 20, 12, 19, 11, 10, 2, 22, 16, 15, 9, 8, 7
29.f	P28	P28	4.75	(2800, 10744)	342.88	90	2, 3, 6, 8, 10, 13, 15, 16, 17, 19, 20, 21, 22, 23, 25, 26, 28	3, 6, 28, 22, 18, 19, 1, 25, 2, 12, 10, 17, 7, 16, 13, 20, 26, 24, 27, 21, 14, 15, 9, 23, 4, 11, 8, 5
30.f	P29	P29	4.75	(3280, 12236)	396.19	91	1, 2, 3, 5, 6, 7, 8, 9, 13, 14, 15, 17, 18, 22, 29	25, 24, 18, 4, 22, 8, 7, 15, 12, 26, 3, 23, 11, 27, 21, 5, 28, 10, 1, 9, 20, 13, 19, 17, 14, 6, 2, 16, 29
31.f	P30	P30	4.75	(3796, 14128)	458	92	2, 3, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 21, 22, 23, 24, 27, 28, 30	4, 9, 14, 7, 27, 1, 19, 13, 23, 24, 3, 5, 11, 22, 29, 30, 25, 6, 21, 10, 17, 15, 18, 20, 28, 12, 2, 8, 26, 16
32.f	P31	P31	4.75	(4372, 16320)	528.25	93	1, 4, 6, 9, 11, 13, 16, 17, 20, 21, 24, 26, 29	9, 8, 23, 7, 2, 14, 22, 1, 19, 15, 24, 31, 17, 5, 3, 25, 30, 10, 26, 11, 13, 28, 12, 6, 27, 4, 21, 18, 20, 29, 16
33.f	P31	P31	4.75	(5044, 18596)	605.81	94	2, 5, 6, 8, 9, 11, 15, 17, 18, 20, 23, 26, 28, 30, 31, 32	27, 7, 8, 2, 3, 9, 6, 19, 13, 28, 24, 21, 15, 32, 11, 1, 26, 14, 12, 4, 29, 23, 31, 10, 22, 20, 5, 25, 16, 17, 30, 18

Table 8: 128-run concatenated designs that optimize the B_4 value. $F_4(128, 112) = (0, 0)$ for designs with 10–15 factors. ^{a}: parent design with best B_4 value; ^{b}: parent with the best F_4 vector; ^{c}: parent with the largest number of degrees of freedom for two-factor interactions.

Design	D_u	D_l	R	$F_5(96, 64, 32)$	B_5	df	γ	δ
10.b	xw.9-3.ac	xw.9-3.ac	5.5	(0, 4, 32)	3	45	1, 2, 3, 4, 6, 7	1, 8, 3, 4, 7, 6, 5, 2, 9
11.b	coa.10.b	coa.10.b	5.5	(0, 2, 88)	6	55	2, 3, 5, 6, 7, 8, 10	1, 2, 5, 4, 3, 8, 7, 6, 9, 10
12.b	xw.11-5.ac	xw.11-5.ac	5.5	(0, 44, 0)	11	66	2, 5, 6, 10, 11	1, 9, 8, 10, 11, 7, 6, 2, 3, 5, 4
13.b	xw.12-6.ac	xw.12-6.ac	5.5	(0, 72, 0)	18	78	6, 9, 10, 12	5, 6, 3, 4, 2, 1, 11, 12, 10, 9, 8, 7
14.b	xw.13-7.ac	xw.13-7.ac	5.5	(0, 112, 0)	28	91	1, 2, 8, 12	1, 2, 3, 6, 7, 4, 5, 12, 13, 10, 11, 8, 9
15.b	xw.14-8.ac	xw.14-8.ac	5.5	(0, 168, 0)	42	105	2, 3, 12, 13, 14	2, 1, 5, 6, 11, 12, 13, 14, 4, 3, 10, 9, 8, 7
Design	D_u	D_l	R	$F_4(128, 112, 96, 80, 64, 48, 32, 16)$	B_4	df	γ	δ
16.b	ma.15-9.1	ma.15-9.1	4	(2, 0, 0, 0, 40, 0, 0, 0)	12	94	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 14	11, 10, 12, 7, 13, 6, 9, 14, 8, 1, 5, 15, 4, 2, 3
17.b	ma.16-10.1	ma.16-10.1	4	(6, 0, 0, 0, 44, 0, 0, 0)	17	99	3, 4, 7, 12, 14, 15, 16	4, 9, 2, 6, 1, 11, 7, 13, 15, 3, 8, 5, 10, 14, 16, 12
18.b	xw.17-11.a	xw.17-11.a	4	(11, 0, 0, 0, 48, 0, 0, 0)	23	99	2, 4, 7, 8, 9, 10, 11, 12, 15, 17	11, 10, 14, 6, 7, 4, 12, 13, 1, 5, 8, 16, 17, 15, 3, 2, 9
19.b	xw.18-12.a	xw.18-12.a	4	(14, 0, 0, 0, 64, 0, 0, 0)	30	100	3, 4, 5, 6, 11, 12, 15, 16, 17, 18	12, 11, 13, 14, 17, 18, 3, 16, 2, 1, 10, 9, 7, 8, 5, 6, 4, 15
20.b	xw.19-13.a	xw.19-13.a	4	(20, 0, 0, 0, 80, 0, 0, 0)	40	105	2, 5, 13, 14, 16, 17, 18	19, 6, 3, 13, 5, 14, 16, 18, 1, 8, 11, 4, 12, 17, 2, 15, 7, 10, 9
21.b	xw.20-14.a	xw.20-14.a	4	(20, 0, 0, 0, 128, 0, 0, 0)	52	106	1, 2, 3, 4, 6, 7, 8, 10, 11, 12, 13, 18, 20	20, 17, 5, 14, 19, 3, 15, 11, 2, 18, 6, 7, 1, 16, 10, 12, 9, 4, 13, 8
22.b	ma.21-15.1	ma.21-15.1	4	(46, 0, 0, 0, 176, 0, 0, 0)	90	83	1, 2, 4, 5, 6, 8, 9, 10, 12, 16, 19, 20, 21	15, 16, 10, 7, 11, 2, 3, 9, 17, 6, 14, 21, 19, 5, 4, 1, 8, 12, 20, 13, 18
23.b	ma.22-16.1	ma.22-16.1	4	(110, 0, 0, 0, 0, 0, 0, 0)	110	83	1, 2, 3, 5, 7, 9, 10, 12, 14, 15, 16, 17, 20	8, 4, 9, 2, 5, 15, 10, 1, 3, 7, 11, 13, 12, 14, 6, 16, 18, 17, 21, 22, 19, 20
24.b	xw.23-17.a	xw.23-17.a	4	(76, 0, 0, 0, 240, 0, 0, 0)	136	85	1, 2, 4, 6, 8, 11, 12, 13, 17, 18, 19, 20, 22, 23	1, 6, 7, 14, 15, 2, 3, 12, 13, 19, 18, 9, 8, 5, 4, 16, 17, 11, 10, 23, 22, 20, 21
25.b	xw.24-18.a	ma.24-18.1	4	(51, 0, 72, 0, 168, 0, 504, 0)	165	86	1, 2, 3, 5, 6, 7, 9, 10, 15, 19, 23	11, 6, 20, 15, 21, 22, 14, 19, 2, 7, 13, 18, 9, 8, 24, 23, 16, 5, 17, 12, 1, 3, 4, 10
26.b	xw.25-19.a	xw.25-19.a	4	(109, 0, 0, 0, 356, 0, 0, 0)	198	87	1, 2, 3, 8, 20, 21, 22, 23, 25	2, 1, 23, 22, 21, 20, 14, 15, 9, 11, 10, 24, 25, 8, 7, 18, 19, 16, 17, 6, 5, 3, 4, 13, 12
27.b	xw.26-20.ac	ma.26-20.1	4	(237, 55, 0, 104, 0, 304, 0, 760)	237	88	1, 2, 3, 4, 5, 16, 17, 18, 21, 23, 24, 25, 26	11, 23, 21, 24, 15, 26, 12, 9, 2, 6, 8, 20, 1, 3, 14, 16, 22, 10, 25, 13, 18, 7, 17, 19, 4, 5
28.b	xw.27-21.ac	ma.27-21.1	4	(69, 0, 120, 0, 352, 0, 888, 0)	280	89	2, 3, 5, 7, 10, 11, 12, 13, 15, 16, 17, 19, 20, 21, 25, 27	10, 23, 19, 22, 18, 14, 5, 27, 24, 12, 6, 4, 15, 1, 11, 25, 7, 16, 21, 17, 13, 20, 26, 2, 8, 9, 3
29.b	xw.28-22.ac	xw.28-22.ac	4	(38, 0, 84, 0, 568, 0, 1644, 0)	330	90	3, 4, 5, 6, 7, 8, 9, 13, 14, 17, 20, 28	2, 10, 4, 23, 17, 9, 13, 26, 27, 19, 22, 11, 14, 25, 7, 15, 6, 12, 1, 28, 21, 20, 3, 24, 16, 8, 18, 5
30.b	xw.29-23.ac	ma.29-23.1	4	(70, 0, 82, 0, 764, 0, 1262, 0)	386	91	2, 3, 6, 7, 11, 12, 17, 19, 20, 21, 23, 24, 26	6, 13, 26, 9, 12, 4, 23, 25, 21, 15, 3, 19, 18, 14, 17, 2, 16, 20, 29, 24, 8, 1, 7, 27, 5, 11, 28, 22, 10
31.b	xw.30-24.ac	xw.30-24.ac	4	(43, 0, 88, 0, 768, 0, 2600, 0)	447	92	4, 5, 6, 8, 9, 10, 12, 13, 15, 16, 17, 24, 26, 30	2, 11, 20, 17, 15, 29, 10, 27, 24, 3, 19, 18, 8, 25, 26, 7, 9, 28, 22, 5, 30, 23, 13, 14, 1, 12, 4, 16, 21, 6
32.b	xw.31-25.ac	ma.31-25.1	4	(60, 0, 84, 0, 1212, 0, 1708, 0)	517	93	1, 2, 4, 5, 6, 8, 10, 11, 13, 15, 16, 17, 18, 19, 24, 26, 27, 30	16, 3, 20, 15, 7, 18, 1, 10, 13, 5, 26, 27, 21, 6, 31, 14, 30, 28, 12, 2, 11, 23, 29, 24, 9, 17, 8, 4, 22, 25, 19
33.b	xw.32-26.ac	xw.32-26.ac	4	(52, 0, 140, 0, 1096, 0, 2996, 0)	592	94	3, 4, 6, 8, 14, 17, 18, 21, 24, 28, 30, 32	23, 5, 16, 3, 20, 22, 8, 18, 30, 2, 27, 32, 6, 12, 24, 21, 7, 14, 15, 25, 4, 17, 19, 29, 26, 1, 11, 10, 13, 28, 9, 31

Paley matrix and our own 64-run concatenated designs as parent designs. A report of all 128-run designs we obtained and their parent designs is given in Sections C and D.

The generalized resolution of the designs we obtained by optimizing the F_4 vector equals 5.25 for 10 and 11 factors and 4.75 for 12–15 factors. When minimizing the B_4 value, all 10–15 factor designs we obtained had a generalized resolution of 5.5. More specifically, our CC/VNS algorithm was able to produce strength-4 designs by minimizing the B_4 value using either the same copy of a 64-run design constructed from quaternary linear codes or one of our own designs that minimize the B_4 value. When minimizing the F_4 vector, our CC/VNS algorithm was able to find strength-4 designs for 10 and 11 factors. These designs, however, have a generalized resolution of only 5.25. So, our CC/VNS algorithm was not able to find designs with a generalized resolution of at least 5.5 when concatenating parent designs based on the folded-over 32-run Paley matrix and sequentially minimizing the F_4 vector.

We compare our strength-4 128-run designs with 10–15 factors with the 128-run designs from Xu and Wong (2007), the regular resolution V designs involving 10 and 11 factors (Xu, 2009) and the minimum G-aberration designs we identified based on the complete enumeration by Schoen et al. (2010). To the best of our knowledge, we are the first to identify the minimum G-aberration 128-run designs. All designs under comparison allow the independent estimation of all two-factor interactions, and have a B_4 value of zero and a zero F_4 vector. For this reason, Table 9 shows the F_5 vector of the designs. For 10 and 11 factors, all tabulated designs are minimum G_2-aberration designs. As a matter of fact, the B_5 values of all 10-factor designs equal 3, while those of all 11-factor designs equal 6. While they are not minimum G-aberration designs, the 10- and 11-factor designs produced by the CC/VNS algorithm have a smaller G-aberration than the corresponding designs of Xu and Wong (2007) and the regular designs of Chen et al. (1993). The minimum G-aberration designs with 10 and 11 factors have a generalized resolution of 5.75, while our designs and those of Xu and Wong (2007) have a generalized resolution of 5.5 only, and the regular designs have a generalized resolution as low as 5. For 12–15 factors, our designs and the designs of Xu and Wong (2007) are minimum G- and G_2-aberration designs.

Table 9: $F_5(128, 96, 64, 32)$ vectors for strength-4 128-run designs with 10–15 factors.

k	CC/B4	QLC	MA	Minimum G-aberration
10	(0, 0, 4, 32)	(0, 0, 12, 0)	(3, 0, 0, 0)	(0, 0, 0, 48)
11	(0, 0, 2, 88)	(0, 0, 24, 0)	(6, 0, 0, 0)	(0, 0, 0, 96)
12	(0, 0, 44, 0)	(0, 0, 44, 0)		(0, 0, 44, 0)
13	(0, 0, 72, 0)	(0, 0, 72, 0)		(0, 0, 72, 0)
14	(0, 0, 112, 0)	(0, 0, 112, 0)		(0, 0, 112, 0)
15	(0, 0, 168, 0)	(0, 0, 168, 0)		(0, 0, 168, 0)

References

Butler, N. A. (2003). Minimum aberration construction results for nonregular two-level fractional factorial designs. *Biometrika*, 90:891–898.

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.

Deng, L.-Y. and Tang, B. (1999). Generalized resolution and minimum aberration criteria for Plackett-Burman and other nonregular factorial designs. *Statistica Sinica*, 9:1071–1082.

Hedayat, A., Sloane, N., and Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications.* Springer.

Schoen, E. D., Eendebak, P. T., and Nguyen, M. V. M. (2010). Complete enumeration of pure-level and mixed-level orthogonal arrays. *Journal of Combinatorial Designs*, 18:123–140.

Sloane, N. J. A. (1999). A library of Hadamard matrices.

Xu, H. (2009). Algorithmic construction of efficient fractional factorial designs with large run sizes. *Technometrics*, 51:262–277.

Xu, H. and Wong, A. (2007). Two-level nonregular designs from quaternary linear codes. *Statistica Sinica*, 17:1191–1213.