DEPARTMENT OF ENGINEERING MANAGEMENT

Two-Level Designs Constructed by Concatenating Orthogonal Arrays of Strenght Three

Alan Vázquez-Alcocer, Peter Goos & Eric D. Schoen

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

City Campus
Prinsstraat 13, B.226
B-2000 Antwerp
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
www.uantwerpen.be

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENGINEERING MANAGEMENT

Two-Level Designs Constructed by Concatenating Orthogonal Arrays of Strenght Three

Alan Vázquez-Alcocer, Peter Goos & Eric D. Schoen

RESEARCH PAPER 2016-011
OCTOBER 2016

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@uantwerpen.be

The research papers from the Faculty of Applied Economics are also available at www.repec.org (Research Papers in Economics - RePEc)

D/2016/1169/011

Two-Level Designs Constructed by Concatenating Orthogonal Arrays of Strength Three

Alan Vázquez-Alcocer[1], Peter Goos[1,2], and Eric D. Schoen[1,3]

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

October 13, 2016

Abstract

Two-level orthogonal arrays of N runs, k factors and a strength of 3 provide suitable fractional factorial designs in situations where many of the main effects are expected to be active, as well as some two-factor interactions. If constructed using the fold-over technique, these designs are called even and allow at most $N/2-1$ interactions to be estimated. For $k < N/3$ factors, there exist strength-3 designs that are not fold-over designs. We call these designs even-odd designs. These designs allow many more interactions to be estimated. For $N \leq 48$, attractive even-odd designs can be extracted from complete catalogs of strength-3 orthogonal arrays. However, for larger run sizes, no complete catalogs exist. In order to construct even-odd designs with $N > 48$, we develop an algorithm for an optimal concatenation of strength-3 designs involving $N/2$ runs. Our approach involves column permutations of one of the concatenated designs, as well as sign switches of the elements of one or more columns of that design. We illustrate the potential of the algorithm by generating two-level even-odd designs with 64, 80, 96, 112 and 128 runs involving up to 33 factors. To the best of our knowledge, our designs with 80 and 96 runs involving fewer than 21 and 25 factors, respectively, are novel, just like our 112-run designs involving up to 29 factors. Our even-odd designs outperform or are competitive with the best known designs in terms of the aliasing of two-factor interactions and in terms of the number of estimable two-factor interactions.

Key words and phrases: Even-odd design, generalized aberration, local search, second-order saturated, two-factor interaction, variable neighborhood search.

1 Introduction

Two-level orthogonal arrays (OAs) of strength 3 can be used to estimate all main effects of $k \leq N/2$ factors in N runs free from two-factor interactions. We denote the two levels for each factor in an OA by -1 and $+1$. Butler (2004, 2007) showed that all strength-3 OAs with $k \geq N/3$ must be even designs. Denoting the generalized word length pattern (GWLP; Xu and Wu, 2001; Ma and Fang, 2001) of a strength-3 design by $(B_4, B_5, \ldots, B_k)$, even designs have a zero B_i value for all odd values of i.

Cheng et al. (2008) showed that a two-level design is even if and only if it is a fold-over design, for which half of the runs are mirror images of the other half. As a result, even designs allow at most $N/2-1$ two-factor interactions to be estimated. For $k < N/3$ factors, there may be strength-3 designs with non-zero B_i values for odd values of i. These designs are called even-odd designs and

generally allow many more two-factor interactions to be estimated. Therefore, they are attractive to experimenters who want to estimate a substantial number of interactions along with the main effects.

Complete catalogs exist of strength-3 two-level OAs with up to 48 runs (Schoen et al., 2010). Based on these catalogs, Schoen and Mee (2012) showed that, for run sizes of 32, 40 and 48, even-odd designs exist for up to 10, 10 and 14 factors, respectively. One way to obtain k-factor even-odd strength-3 designs for which the number of runs is larger than 48 is to concatenate two different strength-3 designs involving $N/2$ runs and $k-1$ factors, which we call parent designs. Subsequently, a factor whose first $N/2$ elements equal -1 and whose last $N/2$ elements are $+1$ can be added to the concatenated design. Designs constructed in this way involve at most $N/4+1$ factors, and their run sizes are multiples of 16. This approach can thus be used to construct even-odd 64-run, 80-run and 96-run designs based on the existing catalogs of strength-3 designs.

Several authors have constructed even-odd designs using variants of the above general approach. Li and Lin (2003), Li et al. (2003) and Cheng et al. (2008) concatenate two copies of a single parent design and subsequently switch the signs of all elements in one or more select columns of the second copy to improve the properties of the concatenated design. Their approach involves a complete enumeration of all possible selections of columns in which to switch the signs of the elements. Li and Lin (2016) suggested to also permute the columns of the second parent design before switching the signs in the selected columns. As a result, this approach also involves an enumeration of column permutations. In any case, the end product of the approaches of Li and Lin (2003), Li et al. (2003), Cheng et al. (2008) and Li and Lin (2016) is a concatenation of a parent design with another design that is isomorphic to that parent design. Two OAs are said to be isomorphic if one array can be obtained from the other array by permuting rows or columns, and switching the signs of the elements in one or more columns. There are two problems with the approaches named. First, it may not be optimal to concatenate two isomorphic designs. Second, for strength-3 designs with run sizes larger than 48, the numbers of factors may be too large to allow for a complete enumeration of all possible sign switches and/or all column permutations. For example, for 17 factors, there are $131,072$ possible choices of columns in which to switch the signs and 3.55687×10^{14} possible column permutations.

The first contribution of this paper is to develop an efficient procedure to construct even-odd designs by concatenating two strength-3 parent designs, which may or may not be isomorphic. More specifically, we propose two interconnected algorithms to find an optimal permutation of the columns of one of the two parent designs, and the best subset of columns for sign switching. Our first algorithm is called the column change (CC) algorithm. It is a local-search-based algorithm making small structured changes to one of the parent designs. This algorithm is embedded in a variable neighborhood search (VNS) algorithm that investigates increasingly diverse new versions of that parent design. Jointly, the two algorithms help to decrease the number of evaluations needed to test all column permutations and all subsets of columns in which to switch the signs. At present, our algorithmic approach requires designs of the same run size and strength, but it can easily be adapted to concatenate designs of different strengths and run sizes.

The second contribution of this paper is to generate two-level concatenated designs with 64, 80, 96, 112 and 128 runs and up to 33 factors. We compare the newly generated designs with the best designs from the literature; see Table 1 for an overview of past and present contributions. In addition to benchmark designs constructed by concatenation, we include the regular 64-run designs reported in Chen et al. (1993), the regular 128-run designs reported in Block and Mee (2005) and Xu (2009), the 64- and 128-run designs constructed using quaternary linear codes by Xu and Wong (2007), and the 64-run designs generated from projections of the folded-over 32-run Hadamard matrix given by the Paley construction, in our comparisons. To our knowledge, our study of the projections from this folded-over 32-run Hadamard matrix is new to the literature.

Table 1: Overview of existing and new two-level designs with $64 \leq N \leq 128$.

N	Reference	k	Parent designs	Technique
64	Chen et al. (1993)	≤ 32	-	complete enumeration of regular designs
	Li and Lin (2003)	≤ 11	regular	partial fold-over
	Xu and Wong (2007)	≤ 32	-	designs from selected quaternary linear codes
	Cheng et al. (2008)	17	strength-3	partial fold-over
	Li and Lin (2016)	≤ 11	resolution IV	concatenation, column permutation, sign switching
	present work	≤ 32	Hadamard matrix	projections of fold-over 32-factor design
	present work	≤ 17	strength-3	CC/VNS, column permutation, sign switching
80	Cheng et al. (2008)	21	strength-3	partial fold-over
	present work	≤ 21	strength-3	CC/VNS, column permutation, sign switching
96	Cheng et al. (2008)	25	strength-3	partial fold-over
	present work	≤ 25	strength-3	CC/VNS, column permutation, sign switching
112	present work	≤ 29	strength-3	CC/VNS, column permutation, sign switching
128	Block and Mee (2005)	≤ 64	-	complete enumeration of regular designs
	Xu and Wong (2007)	≤ 64	-	designs from selected quaternary linear codes
	Xu (2009)	≤ 40	-	complete enumeration of regular designs
	present work	≤ 33	strength-3	CC/VNS, column permutation, sign switching

Our new designs outperform or are competitive with the best known designs in the literature in terms of the aliasing of two-factor interactions and in terms of the number of estimable two-factor interactions. To the best of our knowledge, our designs with 80 and 96 runs and fewer than 21 and 25 factors, respectively, are new to the literature. This is also the case for our 112-run designs involving up to 29 factors.

The rest of this paper is organized as follows. Section 2 presents classification criteria for strength-3 designs. Section 3 describes our algorithmic approach for concatenating two strength-3 parent designs. Section 4 provides an overview of the new designs with $64-128$ runs. We conclude with a discussion and some suggestions for future research.

2 Classification of strength-3 designs

Orthogonal two-level designs are most commonly evaluated in terms of their G-aberration (Deng and Tang, 1999), their generalized resolution, their G_2-aberration (Tang and Deng, 1999), and the rank of the matrix of two-factor interaction contrast vectors. For strength-3 designs, this rank equals the maximum number of two-factor interaction effects that are estimable simultaneously (Cheng et al., 2008). In the rest of this paper, we use the term 'number of estimable two-factor interactions' for

this criterion. The G-aberration criterion ranks designs according to their confounding frequency vectors (CFVs), while the G_2-aberration criterion ranks the designs according to their GWLPs. The generalized resolution is a scalar that quantifies the most severe aliasing between the factorial effects.

The CFV is based on the J_s-characteristics of s-factor interaction contrast vectors. When coding the two levels of each factor as -1 and $+1$, any s-factor interaction contrast vector involves the elements -1 or $+1$. Its J_s-characteristic is the absolute value of the sum of the vector's elements. For strength-3 OAs, any two- or three-factor interaction contrast vector has as many -1s as $+1$s, so that its elements sum to zero. As a result, any J_2- or J_3-characteristic is zero. Deng and Tang (1999) showed that the J_4-characteristics of N-run two-level strength-3 OAs necessarily equal $N - 16q$, where q is a non-negative integer. For instance, for a 64-run strength-3 design, the possible J_4-characteristics are 64, 48, 32, 16 and 0. This implies that the absolute correlations between pairs of two-factor interaction contrast vectors are 1, 3/4, 1/2, 1/4 or 0. A four-factor interaction contrast vector can be calculated as the product of two two-factor interaction contrast vectors. Therefore, whenever J_4-characteristics of 64 occur, this implies that pairs of two-factor interactions are completely aliased. Whenever J_4-characteristics of zero occur, this implies that certain pairs of two-factor interactions are not aliased at all. Intermediate J_4-characteristic values imply partially aliased two-factor interactions.

The frequencies of the J_s-characteristics calculated for all s-factor interaction contrast vectors are generally collected in a vector F_s. The vector's first entry is the frequency of the value N. For strength-3 designs whose run sizes are multiples of 16, the vector's next entries are the frequencies of the J_s-characteristic values $N - 16$, $N - 32$, etc. (Bulutoglu and Ryan, 2015). The frequency of the zero value is usually omitted. The concatenated vector $(F_4, F_5, F_6, \ldots, F_k)$ is the CFV of a strength-3 k-factor design. Note that, for strength-3 orthogonal designs, the F_2 and F_3 vectors are zero vectors, since all J_2- and J_3-characteristics are zero.

As an illustration, the F_4 vector of one of our 64-run 17-factor designs is $F_4(64, 48, 32, 16) = (0, 0, 83, 708)$. So, of all 2380 four-factor interaction contrast vectors, none have J_4-characteristics of 64 or 48, 83 have a J_4-characteristic of 32, and 708 have a J_4-characteristic of 16. The remaining 1589 four-factor interaction contrast vectors have a zero J_4-characteristic and are not explicitly mentioned in the F_4 vector.

The generalized resolution of a design is defined as $s + 1 - \max(J_s)/N$, where s is the size of the smallest subset of factors with a non-zero J-characteristic. So, the generalized resolution of the 17-factor design mentioned above equals 4.5, because (i) there are subsets of four factors with a non-zero J_4-characteristic and (ii) the largest J_4-characteristic equals 32. Therefore, the design involves pairs of two-factor interaction contrast vectors with an absolute correlation of 0.5. More specifically, the design involves 249 of these pairs, where 249 is calculated as the triple of the frequency of 83 with which the J_4-characteristic of 32 occurs.

A k-factor minimum G-aberration design sequentially minimizes the entries of the CFV $(F_4, F_5, F_6, \ldots, F_k)$, from left to right. In this paper, we restrict our attention to the F_4 vector, because we assume three-factor and higher-order interactions to be negligible. Under this assumption, the information contained within the vectors $F_5, F_6, \ldots, F_k$ is not relevant in practice.

A minimum G_2-aberration design sequentially minimizes the GWLP from left to right. The GWLP is a vector of generalized word counts $(B_1, B_2, \ldots, B_k)$, where B_i is the sum of the squared J_i-characteristics divided by N^2. For strength-3 OAs, $B_1 = B_2 = B_3 = 0$, and $B_4 > 0$. For example, the 17-factor design referred to above has a B_4 generalized word count of $0 \times 1^2 + 0 \times (\pm 3/4)^2 + 83 \times (\pm 1/2)^2 + 708 \times (\pm 1/4)^2 = 65$. One way of calculating B_4 values requires the evaluation of all $k!/[4!(k-4)!]$ four-factor interaction contrast vectors. Butler (2003) proposed a computationally cheaper alternative. He expressed the similarity of the runs in a given design D by means of the matrix $T = DD^\top$, and showed that, for any orthogonal array, $B_4 = (M_4 - k(3k-2))/24$, where

$M_4 = N^{-2} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij}^4$ is the fourth moment of T.

3 Concatenation procedure

Our algorithm concatenates two strength-3 orthogonal arrays, D_u and D_l, which both have $N/2$ runs and a given number of factors, say m. Adding the column $\mathbf{z} = [\mathbf{1}_{N/2}^T, -\mathbf{1}_{N/2}^T]^T$ to the concatenated m-factor design generated by our algorithm results in an N-run strength-3 design D with $k = m + 1$ factors. The m two-factor interactions involving the added factor are all orthogonal to the two-factor interactions of the original factors. As a result, the B_4 value and the F_4 vector are not affected by adding the extra factor. Since a strength-3 orthogonal array with $N/2$ runs can accommodate at most $N/4$ factors, our approach yields N-run designs with at most $k = N/4 + 1$ factors. In this respect, our approach is similar to that of Cheng et al. (2008). We refer to D_u, D_l and D as the upper design, the lower design and the concatenated design, respectively.

The first step in the construction process is the selection of two strength-3 orthogonal arrays with $N/2$ runs, which we call parent designs. The parent designs may or may not be isomorphic. One of the parent designs serves as the upper design D_u, while the other becomes the lower design D_l. Which of the two parent designs is the upper and the lower design is arbitrary and does not impact the quality of the resulting concatenated design. Without losing generality, we assume that D_u and D_l are in their lexicographically minimum form (Schoen et al., 2010). Next, we permute the columns of D_l and switch the signs of the elements in a certain number of columns of D_l, so as to improve the concatenated design in terms of a desired criterion. We refer to the lower design produced after switching the signs in a subset of its columns and applying a column permutation as a *plan* for D_l.

Whenever $m \geq N/6$, the $N/2$-run parent designs for our procedure must be even. Sign switches and column permutations in the parent designs do not change their even nature. Therefore, the original m-factor concatenated designs are also even when $m \geq N/6$, and the concatenated designs only become even-odd after adding the column $\mathbf{z}$. It is easy to see that the numbers of estimable two-factor interactions in these designs equal at most $N/2 - 2 + m$. The concatenated designs with added column $\mathbf{z}$ become second order saturated (SOS, Cheng et al., 2008) if the number of main effects, $m + 1$, plus the number of estimable two-factor interactions equals $N - 1$. That can happen only if $N/2 - 2 + m = N - 1 - (m + 1)$, or $m = N/4$. Therefore, the concatenation of even designs with $m \leq N/4$ does not lead to an SOS design, while the concatenation of even designs with $m = N/4$ may or may not lead to an SOS design.

The total number of plans that can be obtained for a lower design D_l by permuting its columns and switching signs in sets of columns is $m! \times 2^m$. Evaluating all possible plans is computationally infeasible when $m \geq 10$. Rather than completely enumerating all plans for the lower design, our concatenation procedure uses a column change (CC) algorithm embedded within a variable neighborhood search (VNS) algorithm. We call the complete concatenation procedure the CC/VNS algorithm.

Our CC/VNS algorithm improves the concatenated design either in terms of the F_4 vector or in terms of the B_4 value. By optimizing the F_4 vector, the CC/VNS algorithm also automatically maximizes the generalized resolution of the concatenated design.

3.1 Column change algorithm

Algorithm 1 shows the CC algorithm, which is a local-search-based algorithm (Michalewicz and Fogel, 2004) that evaluates changes to the current plan for the lower parent design in terms of the B_4 value or the F_4 vector. We developed fast update procedures for the B_4 value and the F_4 vector, so that the evaluation of the B_4 value and the F_4 vector can be done without computing the two-factor interaction matrix from scratch for every change applied by the algorithm. A detailed account of our

update procedures is given in Appendix A. Algorithm 1 requires two m-factor designs with $N/2$ runs as inputs.

The algorithm starts by switching the signs in the leftmost column of the current plan for the lower design D_l and evaluates the resulting concatenated design. If the change does not yield an improvement, the algorithm starts evaluating swaps between the leftmost column and the columns to its right; see lines 11–20 in Algorithm 1. Two types of swaps are performed. The first swap involves the unmodified columns 1 and j, while the second swap involves the original column 1 and the sign-reversed column j. As soon as these modifications to the lower design result in an improvement of the concatenated design, in terms of the B_4 value or the F_4 vector, the algorithm shifts its attention to the second column. First, it switches the signs in column 2 of the current plan for the lower design and evaluates the resulting concatenated design. If the sign switch does not yield a better concatenated design, the algorithm evaluates swaps between column 2 and the columns to its right. This process is repeated for each of the columns in the current plan for the lower design, and it ends with an evaluation of the concatenated design resulting from a sign switch of column m of the current plan for the lower design.

Each time a sign switch of a certain column i or a swap of it with one of the columns to its right results in an improved concatenated design, the algorithm continues its operations on this newly obtained improved design. The algorithm therefore uses a first-improvement optimization strategy. The algorithm makes several passes through all the columns and stops when no better plan can be found for the lower design D_l. The output of Algorithm 1 is an improved plan $D_l^\star$ of the original lower parent design D_l.

3.2 Variable neighborhood search algorithm

Variable neighborhood search or VNS is a metaheuristic introduced by Hansen and Mladenović (2001) as an improvement over local-search-based algorithms for combinatorial optimization. The danger of a local-search-based algorithm is that it may get stuck in a locally optimal solution instead of a global optimum because it does not examine all possible changes to the existing solution. VNS attempts to overcome this weakness of classical local search algorithms by systematically exploring more than one neighborhood structure. A neighborhood structure is defined by a type of change that can be made to a given solution s. Each allowable change is called a move. All solutions s' that can be reached by one move are said to be in the neighborhood $N(s)$ of s. The rationale for using more than one neighborhood is that a solution which is a local optimum with respect to one neighborhood is not necessarily a local optimum with respect to another neighborhood. For this reason, escaping from a locally optimal solution can be done by changing the neighborhood structure. Unlike many other metaheuristics, VNS is simple to implement and requires few, and sometimes even no parameters. Moreover, Hansen et al. (2008) showed that the VNS framework is very general and can be easily extended to integrate features from tabu search (Glover and Laguna, 1997), simulating annealing (Eglese, 1990), and other local search algorithms. VNS has been successfully applied to a wide variety of optimization problems such as vehicle routing (Kytöjoki et al., 2007), project scheduling (Fleszar and Hindi, 2004), automatic discovery of theorems (Caporossi and Hansen, 2004), graph coloring (Avanthay et al., 2003), and synthesis of radar polyphase codes (Mladenović et al., 2003).

On various occasions, VNS has also been used to construct experimental designs. For instance, Garroi et al. (2009) proposed a VNS algorithm to compute D-optimal run orders for central composite designs in the presence of serial correlation. More recently, Sartono et al. (2015) and Syafitri et al. (2015) used VNS to construct fractional factorial split-plot designs and optimal mixture designs in the presence of ingredient availability constraints, respectively.

Our CC/VNS algorithm performs systematic changes to the lower parent design so as to minimize

Algorithm 1: Pseudocode of the column change algorithm.

```
Input: D_u and D_l
D_l* ← D_l
Set i ← 1
repeat
    for i = 1, ..., m do
        Construct plan D_l' by switching signs in column i of D_l*.
        if concatenated design (D_u, D_l') is better than (D_u, D_l*) then
            D_l* ← D_l'
        else
            Set j ← i + 1
            Set no_improvement ← True
            while j ≤ m and no_improvement do
                Construct plan D_l+ by swapping columns i and j of D_l*.
                Construct plan D_l- by switching the signs in column j of D_l* and swapping the
                resulting column with column i.
                Evaluate the concatenated designs (D_u, D_l+) and (D_u, D_l-).
                Set D_l' to be the best of the plans D_l+ and D_l-.
                If both concatenated designs are equally good, select at random.
                if concatenated design (D_u, D_l') is better than (D_u, D_l*) then
                    D_l* ← D_l'
                    no_improvement ← False
                j ← j + 1
until no change in D_l*
Output: Improved plan D_l*
```

the F_4 vector or the B_4 value of the concatenated design. It involves two main components: (i) four neighborhood structures to create neighboring plans from the current best plan of D_l and (ii) the CC algorithm described in Section 3.1 to improve these neighboring plans. Two plans A and B of the lower design are said to be neighboring plans if $A \in N_i(B)$ or $B \in N_i(A)$ for a neighborhood structure N_i. Because of the two main components of our CC/VNS algorithm, this algorithm belongs to the general class of VNS algorithms in which a local search algorithm is used to improve the neighboring solutions created by the neighborhood structures (Hansen et al., 2008).

The four neighborhood structures used by our CC/VNS algorithm are listed in Table 2 and start by modifying one, two, two and three columns, respectively. As a result, the four neighborhoods explore increasingly diverse plans for the lower parent design. Table 2 shows that the size of the first neighborhood structure increases linearly with the number of factors m of the parent designs. For this reason, the size of this neighborhood structure, N_1, is denoted by $O(m)$. The sizes of the second and third neighborhood structures, N_2 and N_3, increase according to a second-order polynomial in m, which is why these sizes are denoted by $O(m^2)$. The size of the last neighborhood, N_4, increases according to a cubic polynomial in m, which is denoted by $O(m^3)$.

The outline of our CC/VNS algorithm is shown in Algorithm 2. The input to the algorithm is an upper parent design D_u and a starting plan for the lower parent design D_l. The starting plan is generated in three steps. First, the signs of all elements in r randomly selected columns of D_l are switched, where r is a random integer between 0 and m. Second, the columns of the resulting plan

Table 2: Neighborhood structures of the CC/VNS algorithm.

N_i	Size	Description
N_1	$O(m)$	Switch signs of any column
N_2	$O(m^2)$	Swap any two columns
N_3	$O(m^2)$	Switch signs of any two columns
N_4	$O(m^3)$	Choose any subset of three columns, move the first two columns one position to the right and move the third column to position 1

are randomly permuted. Third, the resulting plan is optimized by the CC algorithm described in Section 3.1.

The CC/VNS algorithm starts by exploring the first neighborhood structure (N_1) of the starting plan. To this end, it randomly selects a plan from the neighborhood and applies the CC algorithm to it, to attempt to find a better plan for the lower parent design. If a better plan is indeed found, the CC/VNS algorithm continues by exploring the first neighborhood structure of the newly obtained, improved plan. If the CC algorithm does not produce a better plan, a second plan is selected from the first neighborhood structure of the starting plan and an attempt is made to improve it using the CC algorithm. The exploration of the first neighborhood structure of a given plan continues until all plans it contains have been optimized by means of the CC algorithm. If this does not yield any better plan than the current best one, the algorithm starts exploring the second neighborhood structure (N_2), in the same fashion. As soon as the exploration of the second neighborhood structure results in an improved plan, the CC/VNS algorithm returns to the first neighborhood structure and explores that first neighborhood structure of the improved plan. If the exploration of the second neighborhood structure does not produce any improved plans, the third neighborhood structure (N_3) is explored, and, if that does not lead to any improved plans, the fourth neighborhood structure (N_4) is explored. The process is repeated until no further improvement can be reached. In the course of the optimization, each neighborhood structure involves high-quality neighbors of the current best plan for the lower parent design, which has a positive impact on the performance of our CC/VNS algorithm.

Finally, to increase the likelihood of finding a globally optimal plan for the lower parent design, the CC/VNS algorithm is repeated a number of times, each time starting from a randomly generated plan for the lower parent design. This multi-start procedure in the algorithmic construction is common to virtually all design construction algorithms in the literature. Eventually, the overall best plan found for the lower parent design over all iterations is reported.

A Matlab implementation of the CC/VNS algorithm can be obtained from the authors. For specific 32-run, 40-run and 48-run parent designs, we present a comprehensive evaluation of the components (i.e., the neighborhood structures) of the CC/VNS algorithm and the computing times in Appendix B. A key result from our evaluation is that each of the four neighborhoods of the CC/VNS algorithm contributes significantly to the quality of the concatenated designs generated.

3.3 Comparison with a benchmark approach

In this section, we investigate the potential of our CC/VNS algorithm by testing whether it is able to match or even improve the results of Li and Lin (2016, LL16), using the parent designs they used. LL16 constructed 32-run designs with up to 9 factors and 64-run designs with up to 12 factors by concatenating regular resolution IV designs with 16 runs and up to 8 factors and regular resolution IV

Algorithm 2: Pseudocode of the CC/VNS algorithm to improve concatenated orthogonal arrays.

Input: D_u and D_l
$R_l \leftarrow$ generate random plan for D_l
Generate initial plan D_l^* using the CC algorithm and D_u and R_l as input.
Set $i \leftarrow 1$
while $i \leq 4$ **do**
    Set *improvement* $\leftarrow$ False
    repeat
        Randomly select a plan S_l from $N_i(D_l^*)$.
        Generate an improved plan D_l' using the CC algorithm and D_u and S_l as input.
        if *concatenated design* (D_u, D_l') *is better than* (D_u, D_l^*) **then**
            $D_l^* \leftarrow D_l'$
            improvement $\leftarrow$ True
            $i \leftarrow 0$
    until *no unexplored plans left in* $N_i(D_l^*)$ or *improvement*
    $i \leftarrow i + 1$
Output: concatenated design (D_u, D_l^*)

designs with 32 runs and up to 11 factors, respectively. They used the same design as upper parent design and as lower parent design and searched for a plan for D_l that sequentially minimizes the CFV of the concatenated design. LL16 showed that, when the same regular 2^{m-p} design is used as upper and lower parent design, the number of computations required to evaluate all possible sign switches of columns can be reduced from 2^m to 2^p. For parent designs with up to 9 factors, they enumerated and evaluated all $m! \times 2^p$ possible plans for D_l. Evaluating all $m!$ column permutations for parent designs with more than nine factors was computationally infeasible. For this reason, they used a large number of randomly chosen permutations instead of all permutations. They ran their enumeration program for 10- and 11-factor parent designs for 168 hours and reported the best results found. They did not report the computing times for the complete enumeration of concatenated designs based on parent designs with up to 9 factors.

We base our evaluation of the CC/VNS algorithm on the results we obtained after 1,000 iterations. These results are shown in Table 3. The first and second columns of the table show the run size N of the concatenated design and the number of factors m of the parent designs, respectively. The third column is the ID of the parent design, as given in Chen et al. (1993). The fourth column presents the frequency of the J_4-characteristic of 16 ($F_4(16)$) for the best 32-run designs we found and the frequency of the J_4-characteristic of 32 ($F_4(32)$) for the best 64-run designs we found. For all 32-run designs we obtained, $F_4(32) = 0$, meaning that J_4-characteristics of 32 do not occur. For all 64-run designs we obtained, $F_4(64, 48, 16) = (0, 0, 0)$, meaning that J_4-characteristics of 64, 48 and 16 do not occur. The F_4 vector of the best designs we obtained coincide with those found by LL16, except for the 11-factor 64-run design based on parent design 11-6.2. The design produced by our CC/VNS algorithm outperforms the benchmark design of LL16, for which $F_4(32) = 46$. For our design, $F_4(32) = 44$.

The fifth column of the table shows the percentage of iterations during which the CC/VNS algorithm was able to find the best F_4 vector. For all but three of the cases in Table 3, this percentage equals 100. In other words, the CC/VNS algorithm generally obtained the best possible concatenated design at each iteration. For the parent designs 7-2.1 and 10-5.4, the percentages were 88.1 and 96.8, respectively. For the 12-factor 64-run design constructed from parent design 11-6.2, the CC/VNS algorithm produced the best concatenated design in 65.9% of the iterations. In the remaining 34.1%

Table 3: Results produced by the CC/VNS algorithm for 32-run and 64-run concatenated designs constructed from regular resolution IV designs. The labels of the parent designs are those used by Chen et al. (1993). The number of plans evaluated to find the optimal F_4 vector is expressed as a percentage of the total number of possible plans averaged over 1,000 iterations.

N	m	Parent	$F_4(16)$ or $F_4(32)$	Percentage of iterations	Percentage of all plans
32	6	6-2.1	4	100	134.104
	7	7-3.1	12	100	18.953
	8	8-4.1	24	100	2.132
64	7	7-2.1	0	88.1	24.580
		7-2.2	0	100	27.421
		7-2.3	4	100	39.416
	8	8-3.1	4	100	4.643
		8-3.2	6	100	4.765
		8-3.3	8	100	4.649
		8-3.4	12	100	4.492
	9	9-4.1	8	100	0.448
		9-4.2	12	100	0.441
		9-4.3	12	100	0.464
		9-4.4	16	100	0.444
		9-4.5	24	100	0.419
	10	10-5.1	16	100	0.036
		10-5.2	24	100	0.036
		10-5.3	26	100	0.037
		10-5.4	30	96.8	0.044
	11	11-6.1	42	100	0.002
		11-6.2	44	65.9	0.003

of the iterations, the CC/VNS algorithm produced a concatenated design with the same F_4 vector as the one found by LL16.

Column 6 of Table 3 shows the number of plans for the lower parent design D_l explored by the algorithm, relative to $m! \times 2^p$ and expressed as a percentage. In all but one case, less than 40% of the total number of plans are evaluated to reach the final result. The exception is the parent design 6-2.1, for which the CC/VNS algorithm evaluated 34% more plans than a complete enumeration would ($6! \times 2^2 = 2880$). Remarkably, a close inspection of our computational results revealed that, for this case, the CC algorithm provided a starting plan with an optimal F_4 vector in all iterations of the CC/VNS algorithm. Constructing this plan only required 2.35% of the total number of evaluations. The additional computations are due to successive visits to the four neighborhood structures by the CC/VNS algorithm, to confirm the good quality of the starting solution produced by the CC algorithm.

From our comparison with the benchmark approach and the comprehensive evaluation of the CC/VNS algorithm in Appendix B, we conclude that our CC/VNS algorithm is successful in creating high-quality concatenated designs using considerably less computing effort than needed by LL16. Encouraged by these results, we used our algorithm to generate two-level even-odd designs for run sizes 64, 80, 96, 112 and 128, which can accommodate up to 33 factors.

4 Results

In this section, we discuss the details of the concatenated designs with 64, 80, 96, 112 and 128 runs and up to 33 factors produced by our CC/VNS algorithm based on selected parent designs with 32, 40, 48, 56 and 64 runs. A detailed description of the parent designs is included in Appendix C. Detailed tables with properties of the concatenated designs are presented in Appendix D.

For each combination of number of runs and number of factors, we considered several pairs of attractive parent designs. If we denote the number of different parent designs considered for each scenario by r, then the total number of concatenated designs we constructed is $r(r+1)/2$. In other words, we concatenated each pair of attractive parent designs to investigate which pair gives rise to the best concatenated design. For each given pair of parent designs, we used 40 iterations of the CC/VNS algorithm when the objective was to minimize the B_4 value and 10 iterations when the objective was to sequentially minimize the F_4 vector. We used a larger number of iterations when minimizing the B_4 value because calculating the B_4 value is computationally less demanding than calculating the F_4 vector. After running the CC/VNS algorithm, we constructed our final k-factor design by adding column $\mathbf{z}$ to the resulting m-factor concatenated design. Appendix D shows that 23 of our best 168 concatenated designs are constructed from different parent designs.

We compare our concatenated designs to the regular resolution IV designs of Chen et al. (1993), Block and Mee (2005), and Xu (2009), to the designs of Xu and Wong (2007) based on quaternary linear codes, to the designs of Cheng et al. (2008) based on partial fold-over, and to the best projections of the folded-over 32-run Hadamard matrix given by the Paley construction. Chen et al. (1993) enumerated all regular designs with 64 runs, while Block and Mee (2005) enumerated all even-odd regular designs with 128 runs. Xu (2009) enumerated regular designs with 128–1024 runs. His 128-run designs are identical to those of Block and Mee (2005) for fewer than 40 factors. For completeness, in our comparisons, we included all regular designs with a minimum B_4 value. Using quaternary linear codes, Xu and Wong (2007) constructed two-level designs with run sizes equal to 64 and 128, and reported the best designs in terms of the G- and G_2-aberration criteria. Cheng et al. (2008) presented 64-, 80-, and 96-run designs with 17, 21 and 25 factors, respectively, that maximize the number of estimable two-factor interactions. The projections from the 32-run Hadamard matrix given by the Paley construction are known to have a generalized resolution of 4.75 for up to 32 factors. In the following, we refer to the 32-run Hadamard matrix resulting from the Paley construction as the 32-run Paley matrix.

4.1 64 runs

Figure 1 shows the generalized resolution, the generalized word count of length 4 (i.e., the B_4 value) and the number of estimable two-factor interactions (denoted by df(2FIs)) of 64-run designs involving 9–17 factors, along with the $F_4(32)$ values for designs with 11–14 factors and the $F_4(64)$ values for designs with 16 and 17 factors. The results reported in the figure are for the best concatenated designs in terms of the F_4 vector and the B_4 value as well as for the benchmark designs of Chen et al. (1993), Xu and Wong (2007) and Cheng et al. (2008). Additionally, we show results for the best projections of the folded-over 32-run Paley matrix. As explained in Appendix C, the parents designs for the 64-run concatenated designs were obtained from the enumeration of Schoen et al. (2010).

Figure 1a shows that the best designs in terms of generalized resolution are derived from the folded-over 32-run Paley matrix and that, for 9–11 factors, the CC/VNS algorithm produced designs with the same generalized resolution. For 9 and 10 factors, this happened regardless of the optimization criterion used by our CC/VNS algorithm. For 11 factors, the generalized resolution of 4.75 was only attained by the CC/VNS algorithm when sequentially minimizing the F_4 vector. The 9-, 10- and

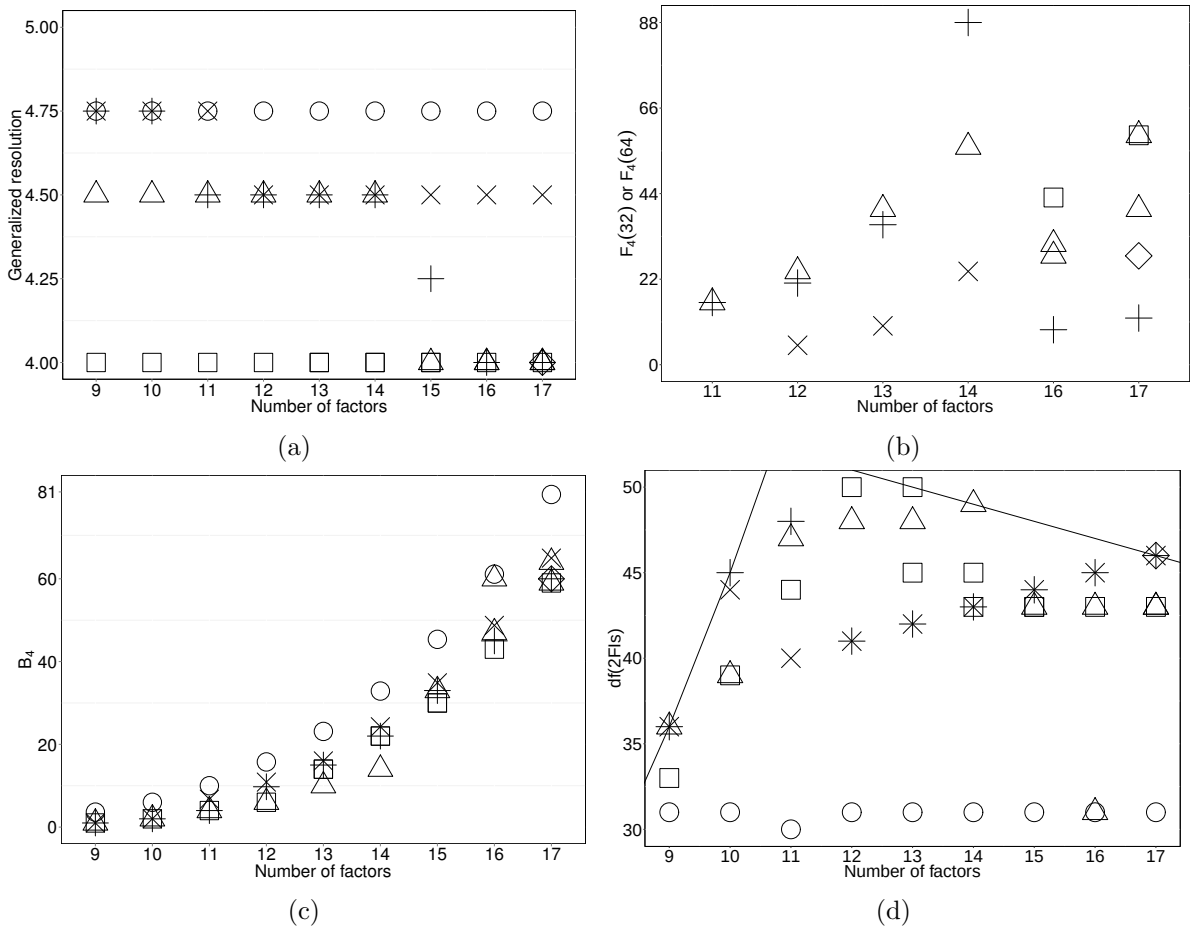


Figure 1: Properties of alternative designs with 64 runs and 9–17 factors. Squares: regular resolution IV designs (Chen et al., 1993); triangles: designs based on quaternary linear codes (Xu and Wong, 2007); diamond: the 17-factor design of Cheng et al. (2008); circles: projections of the folded-over 32-run Paley matrix; crosses: designs produced by the CC/VNS algorithm using the F_4 vector as optimization criterion; pluses: designs produced by the CC/VNS algorithm using the B_4 value as optimization criterion. Lines in (d): maximum number of estimable two-factor interactions.

11-factor designs with generalized resolution 4.75 produced by the CC/VNS algorithm have $F_4(16)$ values of 16, 32 and 108, respectively, while the projections from the folded-over 32-run Paley matrix have $F_4(16)$ frequencies of 58, 96, and 160, respectively. So, the 9-, 10- and 11-factor designs produced by the CC/VNS algorithm perform better in terms of G-aberration, and also in terms of the B_4 value and in terms of the G_2-aberration criterion. Because the 9-, 10- and 11-factor designs produced by the CC/VNS algorithm are even-odd designs, they allow a larger number of two-factor interactions to be estimated than the designs obtained from the 32-run Paley matrix (which is even by construction, and allows at most 31 interactions to be estimated). This is clearly shown in Figure 1d. As a result, the 9-, 10- and 11-factor designs produced by our CC/VNS algorithm outperform the projections from the folded-over 32-run Paley matrix.

For 11, 12, 13, 14, 16 and 17 factors, at least one of the designs generated by our CC/VNS algorithm has the same generalized resolution as the designs constructed using quaternary linear codes, the regular designs, and the SOS design of Cheng et al. (2008). In Figure 1b, we compare these designs in terms of their F_4 vector. More specifically, the figure shows the $F_4(32)$ frequencies of the

designs with 11–14 factors and a generalized resolution of 4.5, while it shows the $F_4(64)$ frequencies of the designs with 16 and 17 factors with a generalized resolution of 4. The 11-factor designs under comparison have the same $F_4(32)$ frequencies. For 12, 13, 14, 16 and 17 factors, the CC/VNS designs under comparison have better F_4 vectors than the other designs, except for the 14-factor concatenated design with optimized B_4 value.

Figure 1c shows that the four 9- and 10-factor designs produced by the CC/VNS algorithm and the 11-factor one minimizing the B_4 value possess the smallest B_4 value of all designs discussed here, just like the regular designs and those constructed using quaternary linear codes. For 9 and 10 factors, the B_4 values we obtained are the minimum values possible (Xu, 2005). For 12–17 factors, the B_4 values of the designs produced by the CC/VNS algorithm are smaller than those of the designs obtained from the folded-over 32-run Paley matrix, but larger than those of the regular designs and the designs constructed using quaternary linear codes. The 17-factor 64-run design of Cheng et al. (2008) has the same B_4 value as our 64-run design obtained by minimizing the B_4 value, but a smaller B_4 value than the one we obtained by sequentially minimizing the F_4 vector.

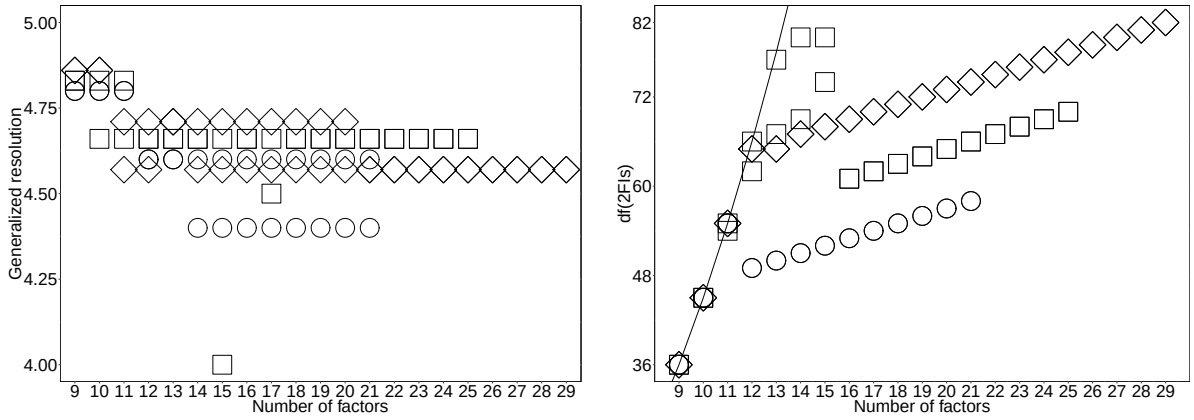
Figure 1d shows that the CC/VNS algorithm produces designs with 9–11 and 15–17 factors that allow at least as many two-factor interactions to be estimated as the best benchmark designs. For 12–14 factors, the numbers of estimable interactions of the designs produced by the CC/VNS algorithm are larger than those of the designs based on the folded-over 32-run Paley matrix, but smaller than those of the regular designs and those of the designs constructed using quaternary linear codes. Note that the even designs constructed from projections of the folded-over 32-run Paley matrix can only estimate up to 31 two-factor interactions. More information on how the projections from the folded-over 32-run Paley matrix were selected is provided in Appendix C. All 32-run parent designs with more than $m = 10$ factors are even. The CC/VNS algorithm succeeded in identifying a plan for the lower parent designs for these cases such that the numbers of estimable two-factor interactions equal $N/2 - 2 + m$, which is the maximum number possible.

The lines in Figure 1d visualize the number of estimable two-factor interactions for each number of factors. The left line corresponds to cases in which all two-factor interactions are estimable, while the right line corresponds to SOS designs. The left line shows that our CC/VNS algorithm is capable of producing designs which allow all interactions to be estimated when there are 9 or 10 factors. The right line shows that our best 17-factor design, the 17-factor design of Cheng et al. (2008), the regular resolution IV design with 13 factors and the 14-factor design of Xu and Wong (2007) are SOS designs.

4.2 80, 96 and 112 runs

To the best of our knowledge, almost all our designs with 80, 96, and 112 runs are new to the literature. For these run sizes, hardly any benchmark designs are available. We obtained the 40-run parent designs for the 80-run concatenated designs from the complete catalog of OAs of Schoen et al. (2010). The 48-run parent designs for the 96-run concatenated designs were those recommended by Schoen and Mee (2012). In the absence of complete catalogs of 56-run OAs, we generated the parent designs for the 112-run concatenated designs from the best projections of folded-over 28-run OAs with 27 factors. Schoen et al. (2015) showed how to generate the complete catalog of 27-factor 28-run OAs from 28-run Hadamard matrices. A detailed description of the selection of the parent designs and the best projections from the 28-run Hadamard matrices is included in Appendix C. In Figure 2, we show the generalized resolution (Figure 2a) and the number of estimable two-factor interactions (Figure 2b) of our designs.

The designs with the highest generalized resolution for 11 and 21–25 factors are 96-run designs rather than 112-run designs. A partial explanation for this result is in the generalized resolution of the parent designs. For the 96-run designs, the parents all have a generalized resolution of 4.67. For



(a) (b)

Figure 2: Properties of alternative designs with 80 runs (circles), 96 runs (squares) and 112 runs (diamonds) produced by the CC/VNS algorithm.

the 112-run designs, the parent designs have a generalized resolution of 4.57 or less. Therefore, in terms of generalized resolution, the parent designs for the 112-run designs are inferior to those for the construction of the 96-run concatenated designs. For up to 20 factors, however, the CC/VNS algorithm managed to combine the two parent designs in such a way that the final concatenated designs no longer involve J_4-characteristics of 24, as a result of which the generalized resolution of the concatenated 112-run design exceeds 4.67 and the 112-run designs are better than the 96-run designs in terms of generalized resolution. For cases with more than 20 factors, the CC/VNS algorithm was unable to avoid J_4-characteristics of 24 in the final concatenated designs with 112 runs.

It is interesting to point out that the 64-run designs involving 12–29 factors, which are based on projections of the folded-over 32-run Paley matrix, have a better generalized resolution than the corresponding designs in Figure 2a. Of course, a major disadvantage of the latter designs is that they allow at most 31 two-factor interactions to be estimated and have larger B_4 values.

The line in the left part of Figure 2b identifies designs in which all two-factor interactions can be estimated. For 9 and 10 factors, there are also 64-run designs with this attractive property (see Figure 1d). For 11 factors, both 80-run designs, both 112-run designs and one of the 96-run designs produced by our CC/VNS algorithm allow the estimation of all the interactions. The 96-run designs with 12–15 factors perform extremely well in terms of the number of estimable interactions. One of the two 12-factor 96-run designs allows all interactions to be estimated, while the 13-factor 96-run design allows all but one interaction to be estimated. The 14- and 15-factor 96-run designs allow 80 interactions to be estimated. For more than 15 factors, the 96-run designs are inferior to the 112-run designs in terms of the number of estimable interactions. For most numbers of factors, both optimization criteria produced designs with the same number of estimable two-factor interactions, but, sometimes, with different generalized resolutions.

The tables in Appendix D show that whether we use the F_4 vector or the B_4 value as the objective in our CC/VNS algorithm almost has no impact on the B_4 value of the concatenated designs. As might be expected, for any given number of factors, the B_4 values decrease with the run size, indicating that larger designs involve less severe aliasing. Our 112-run designs with 9–15 factors, however, have larger B_4 values than the corresponding 96-run designs.

As the designs of Cheng et al. (2008), our 21-factor 80-run designs and our 25-factor 96-run designs are SOS designs. Therefore, they allow 58 and 70 two-factor interaction effects to be estimated. In

Table 4: Properties of the 80-run designs with 21 factors and the 96-run designs with 25 factors produced by the CC/VNS algorithm and by Cheng et al. (2008).

N	k	Design	Generalized resolution	$F_4(96, 80, 64, 48, 32, 16)$	B_4
		Cheng et al. (2008)	4.2	(0, 0, 125, 0, 288)	136.52
80	21	CC/VNS (B_4)	4.2	(0, 0, 4, 242, 2320)	132.96
		CC/VNS (F_4)	4.4	(0, 0, 0, 216, 2557)	136.84
		Cheng et al. (2008)	4	(30, 0, 0, 0, 1940, 0)	245.55
96	25	CC/VNS (B_4)	4.67	(0, 0, 0, 0, 861, 5240)	241.22
		CC/VNS (F_4)	4.67	(0, 0, 0, 0, 708, 5984)	244.89

Table 4, we compare our designs to the benchmarks in terms of the F_4 vector and in terms of the B_4 value. The F_4 vectors in the upper part of Table 4 show that the two 21-factor 80-run designs we obtained using the CC/VNS algorithm have a smaller G-aberration than the design of Cheng et al. (2008). The 21-factor 80-run design produced by the CC/VNS algorithm by minimizing the B_4 value outperforms the design of Cheng et al. (2008) in terms of the B_4 value. The lower part of Table 4 (more specifically, the $F_4(96)$ value of 30) indicates that the 25-factor 96-run design of Cheng et al. (2008) involves some complete aliasing between certain pairs of two-factor interactions. It therefore has a generalized resolution of 4 only. In contrast, the two designs produced by the CC/VNS algorithm do not involve completely aliased two-factor interactions. Their largest J_4-characteristics equal 32, so that they both have a generalized resolution of 4.67. We conclude that the CC/VNS algorithm produced designs that are very attractive compared to those of Cheng et al. (2008).

4.3 128-run designs

It is known that there exist strength-4 128-run designs 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 the concatenated 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 Paley matrix and our own 64-run concatenated designs, as parent designs. A detailed report of all 128-run designs we obtained and their parent designs is given in Appendix 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 designs that minimize the B_4 value. When minimizing the F_4 vector, our CC/VNS algorithm was able to find strength-4 designs only 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

Table 5: $F_5(128, 96, 64, 32)$ vectors for strength-4 128-run designs with 10–15 factors.

k	CC/VNS (B_4)	Xu and Wong (2007)	Regular Resolution V	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)

and Wong (2007), the regular resolution V designs in 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 5 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. 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.

Figure 3 presents the most important properties of the 128-run designs we constructed with 16–33 factors, by minimizing the B_4 value and sequentially minimizing the F_4 vector. The figure also shows the properties of the designs of Xu and Wong (2007) and the regular designs identified by Block and Mee (2005).

Figure 3a shows that the designs we obtained by sequentially minimizing the F_4 vector outperform all benchmark designs, as well as all designs we obtained by minimizing the B_4 value, in terms of generalized resolution. Their generalized resolution is 4.75. Since the parent designs also had a generalized resolution of 4.75, this implies that the concatenation algorithm was unable to improve the generalized resolution. It must be pointed out, however, that the F_4 vectors of the concatenated designs are substantially better than the F_4 vectors of the parent designs. This is shown in the appendix. Figure 3a also shows that the designs we obtained by minimizing the B_4 value all have a generalized resolution as low as 4. Despite their low generalized resolution, the designs with 21, 25 and 28–33 factors have a smaller G-aberration than the benchmark designs. This can be seen from the $F_4(128)$ values displayed in Figure 3b.

Figure 3c shows the B_4 values of the 128-run designs we constructed. For none of the cases, our designs have B_4 values as low as those of the benchmark designs. Most likely, this is due to the limited number of parent designs we used, and to their quality. To the best of our knowledge, a complete catalog of strength-3 64-run designs is not available, so that we were unable to select better parent designs.

Finally, Figure 3d shows that we were able to find designs with 20, 21, 30, 31, 32 and 33 factors which allow more two-factor interactions to be estimated than the benchmark designs. The line in the figure identifies SOS designs, so the two 33-factor designs we obtained and one of our 21-factor designs

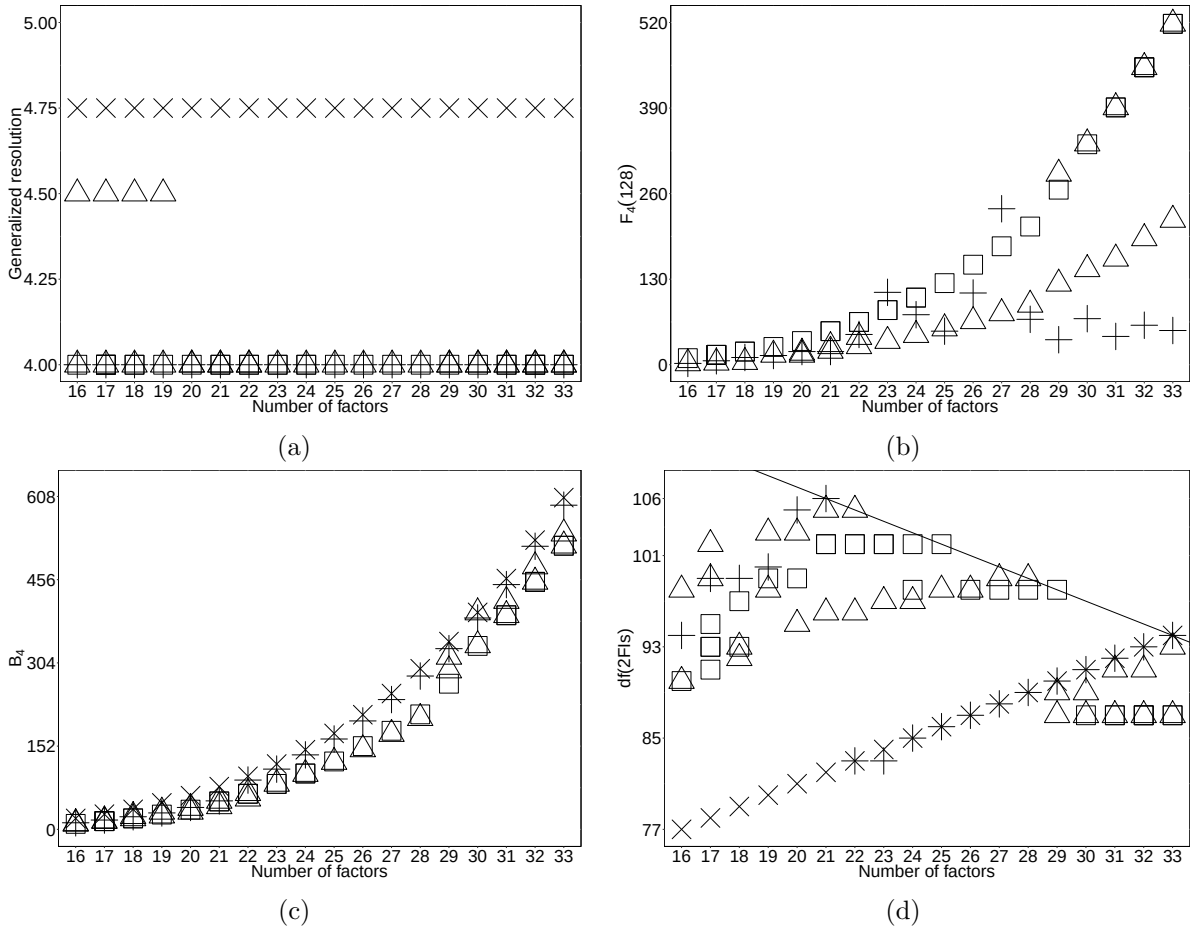


Figure 3: Properties of alternative designs with 128 runs. Squares: regular resolution IV 128-run designs (Block and Mee, 2005); triangles: designs obtained from quaternary linear codes (Xu and Wong, 2007); crosses: designs produced by the CC/VNS algorithm using the F_4 vector as optimization criterion; pluses: designs produced by the CC/VNS algorithm using the B_4 value as optimization criterion. Line in (d): maximum number of estimable two-factor interactions.

are SOS designs. The same goes for the 28-factor design of Xu and Wong (2007) and the regular resolution IV designs with 25 and 29 factors.

5 Discussion

In this paper, we introduced the CC/VNS algorithm to optimize the concatenation of two strength-3 orthogonal arrays. The algorithm employs sign switches and column permutations to minimize the aliasing among two-factor interactions of the concatenated design. Using the CC/VNS algorithm, we generated two-level even-odd designs with 64, 80, 96, 112 and 128 runs with up to 33 factors. Our even-odd designs outperform or are competitive with the best known designs in the literature in terms of the aliasing of two-factor interactions and in terms of the number of estimable two-factor interactions.

More importantly, the CC/VNS algorithm provides a fast, flexible, and efficient alternative to more specialized constructions and complete enumerations that only work for a limited number of scenarios.

The run sizes that can be handled with our approach, which are multiples of 16, are more flexible than the run sizes handled by Chen et al. (1993), Block and Mee (2005), Xu and Wong (2007), and Xu (2009), which are powers of two. This flexibility allowed us to create two-level designs with 80 and 96 runs and up to 21 and 25 factors, as well as 112-run designs involving up to 29 factors that, to the best of our knowledge, are new to the literature. These designs fill the large gap between the run sizes of 64 and 128, and provide useful options to reduce the aliasing among two-factor interactions whenever 9–25 factors are studied. When compared to a complete and exhaustive search of partial fold-over plans like in Cheng et al. (2008), our CC/VNS algorithm produced better designs within seconds; see Appendix B.

All but one 64-run design created by our CC/VNS algorithm have a smaller G-aberration than the designs of Chen et al. (1993), Block and Mee (2005), Xu and Wong (2007), and Xu (2009). We obtained the best 64-run designs in terms of G-aberration from projections from the folded-over 32-run Hadamard matrix given by the Paley construction. A drawback of these designs is that they are even. Therefore, they allow at most 31 two-factor interactions to be estimated. The even-odd 64-run designs we obtained by sequentially minimizing the F_4 vector for 9–11 factors have a better G-aberration than those based on the folded-over Paley matrix.

The 128-run designs we obtained by sequentially minimizing the F_4 vector for 16–33 factors have a better generalized resolution than all alternative designs available from the literature. We recommend the use of these designs when the experimenter's interest is in minimizing the correlation between pairs of two-factor interaction contrast vectors. Despite the fact the CC/VNS algorithm did not optimize the F_5 vector or the B_5 value, we produced strength-4 128-run designs that are competitive with the best designs from the literature. Note that our CC/VNS algorithm can easily be modified to optimize the F_5 vector or the B_5 value, or other criteria such as the generalized resolution or the number of estimable two-factor interactions.

Some of our 112-run designs have a smaller generalized resolution than our 96-run designs with the same number of factors. Similarly, the 128-run designs with 10–15 factors we obtained by optimizing the F_4 vector have a smaller generalized resolution than the designs we obtained by optimizing the B_4 value. This suggests that the projections of the folded-over 28-run or 32-run orthogonal arrays were not ideal for constructing concatenated designs that optimize the F_4 vector. More research on strength-3 56- and 64-run designs is needed to improve the parent designs required for constructing 112- and 128-run concatenated designs.

The extra column $\mathbf{z}$ with $N/2$ plus ones and $N/2$ minus ones causes even concatenated designs, produced by concatenating two even parent designs, to become even-odd, and, for $k = N/4 + 1$ factors, SOS. When based on even-odd parent designs, concatenated designs can be even-odd without the inclusion of this column. Even-odd and SOS designs are attractive for estimating models including all the main effects and a considerable number of two-factor interactions. Model selection strategies for this scenario can be found in Draguljić et al. (2014), for instance. Alternatively, on might conduct a two-stage analysis similar to the one proposed by Miller and Sitter (2001). In the first stage of their proposed analysis, the active main effects are identified, while in the second stage only two-factor interactions obeying effect heredity are studied. An interesting subject for further research is to improve this approach by taking into account the independence of the two-factor interactions involving the extra column $\mathbf{z}$.

The column $\mathbf{z}$ can also be used as a blocking factor to arrange the concatenated design in two blocks of size $N/2$. This blocking factor is orthogonal to the main effects and to all second-order interactions of the remaining factors. Thus, the upper parent design D_u and the optimal plan for the lower parent design D_l can be run on two different days, machines, etc., offering more flexibility for the experimentation.

Finally, the CC/VNS algorithm may be able to improve on the designs with 64 and 128 runs of Xu and Wong (2007) in terms of G_2-aberration, by concatenating strength-2 parent designs instead of strength-3 parent designs. The main challenge here is to identify the ideal strength-2 parent designs. This would be an interesting topic for future research. Since our algorithmic approach is very general, it would also be interesting to investigate the concatenation of orthogonal arrays of different strengths and even different sizes. In addition, the parent designs considered could include nonorthogonal arrays, multi-level arrays or mixed-level arrays.

Appendices

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 these scalar and vector quantities, 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 $k!/\left[4!(k-4)!\right]$ combinations of 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 $T = DD^\top$, 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 uses the following objective function to improve the B_4 value of the concatenated design:

$$b(D) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}^4, \text{ and } C = D_u D_l^T.$$

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 $[2m-1]\,n^2$. So, every time the algorithm sign switches a column or swaps two columns in D_l, recalculating the matrix $D_u D_l^T$ requires at least $[2m-1]\,n^2$ calculations to get the resulting objective value. A computationally cheaper approach is to change only the elements in C that correspond to the columns in 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 as $C' = C - 2\, u_i\, v_i^T$, 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 up 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 the 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 construct matrices $-u_i\, v_i^T, -u_j\, v_j^T, u_i\, v_j^T$ 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 of 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, or 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 required by each procedure suggests a clear preference for our updating procedure, our Matlab implementation only include the updating formula for a sign switch in the lower design. A computing time study (not shown) revealed that the updating formula for a swap of two columns in D_l requires the same or slightly more time than just changing the positions of the two columns and calculating matrix C from scratch. For this reason, we used the latter approach in our Matlab implementation of the CC/VNS algorithm. Finally, our updating procedure can be extended to deal with the sign-reversal of more than one column and changing the positions of more than two columns with possible sign switches. We believe that our updating procedure could be useful for alternative implementations of our algorithm.

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, $r = N/16 - 1$. 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 $M_0 >> \cdots >> M_r$. The CC/VNS algorithm uses $M_i = 10^{5(r-i)}$, where $i = 0, 1, \ldots, r$.

For some programming languages like 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 $c_i \odot c_j$, $i = 1, \ldots, m-1$ and $j = i+1, \ldots, m$; where c_i is the i-th column of D_u (D_l). 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 occur 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 up 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

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

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 when 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 given by 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 involves v_i or v_j.

Table A1: Design cases 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 in an element of the F_4 vector means that no J_4-characteristics 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

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 A1 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 OAs with different F_4 vectors. The last instance, OA96Two, is based on two identical minimum G-aberration OAs.

Table A2 shows the computing time required for a completed optimization by the CC/VNS algorithm. For each of the cases in Table A1, Table A2 gives the averages and standard deviations of 10 optimizations split according to the objective function to be 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 objective function F_4 than to minimize B_4. For the 96-run design cases with $m = 24$ factors, more than one hour is needed for a single optimization. To construct designs that optimize the F_4 vector with run sizes $N > 96$ and $m > 24$, we have to restrict the size of the neighborhood N_4 to be $24!/\,[3!(24-3)!] = 2024$, the size of N_4 when $m = 24$, to keep the computing times for one iteration within 2 hours.

Table A2: 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

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 A1, 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 each of which is found by searching through 10,000 random concatenated designs.

Figure A1 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 A1a 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 will be above the median. For the random search, the median coincides with the lower quartile, so that few of the B_4 values produced by the random search will be below the median. This shows that we are better off by using the CC algorithm than by conducting a random search.

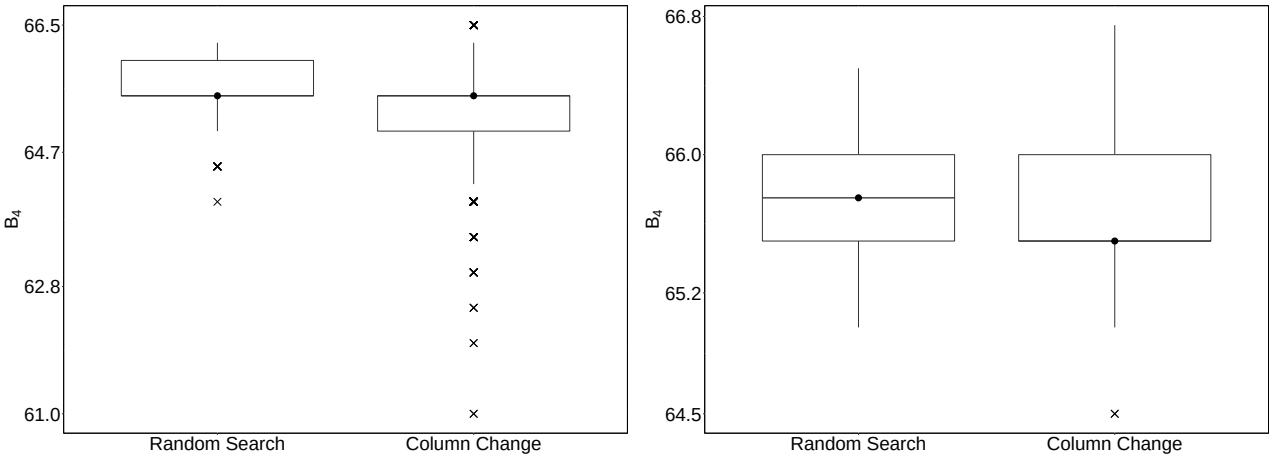
The case of OA64Two is illustrated in Figure A1b. Here, the median of the B_4 values provided by the CC algorithm is smaller than the median of the B_4 values of the random search. Finally, for the larger cases, Figures A1c−A1e clearly show that the majority of the B_4 values obtained by the CC algorithm is 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 A2 shows the distribution of the generalized resolutions of the concatenated designs from the random search and the CC algorithm. Figure A2a 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 A2b shows that the CC algorithm and the random search resulted in the same number of concatenated designs with resolution 4.25 and with resolution 4.5. For the 80-run case, Figure A2c shows that the CC algorithm only produced concatenated designs with a generalized resolution of 4.6, where the random search generated 85 designs with a generalized resolution of 4.25. 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 A2d. 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 A2 does not include a separate panel for the OA96Two case.

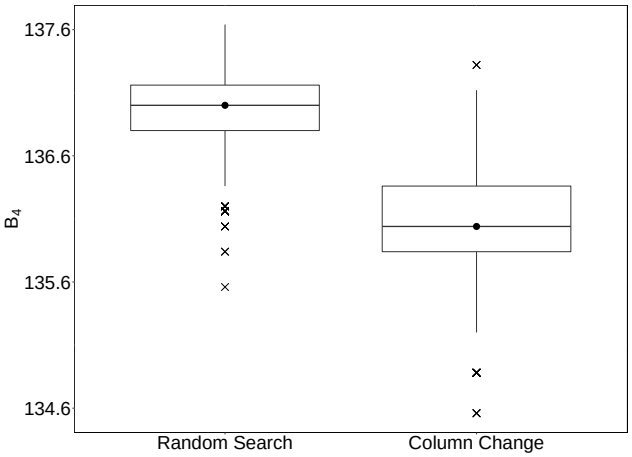
We now turn to the $f(D)$ value of the designs that optimize the F_4 vector. Recall from Section A.2 that $f(D)$ is a linear combination of the elements of the F_4 vector, where the frequencies of large J_4-characteristics receive a larger weight than those of small J_4-characteristics. Low values for 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.

Figure A3 shows the $f(D)$ values for the concatenated designs from the random search and the CC algorithm. Figure A3b and Figure A3c include only designs with generalized resolution of 4.5 and 4.6, respectively, because otherwise, the weights for the large J_4-characteristics would distort the figure. Figure A3a 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 will be above the median. For the random search, the median coincides with the lower quartile, so that few of the $f(D)$ values will be below the median. So we are better off by using the CC algorithm than by conducting a random search. Figures A3b–A3e for the other design cases show that the CC algorithm generated concatenated designs with smaller F_4 vectors than the random search.

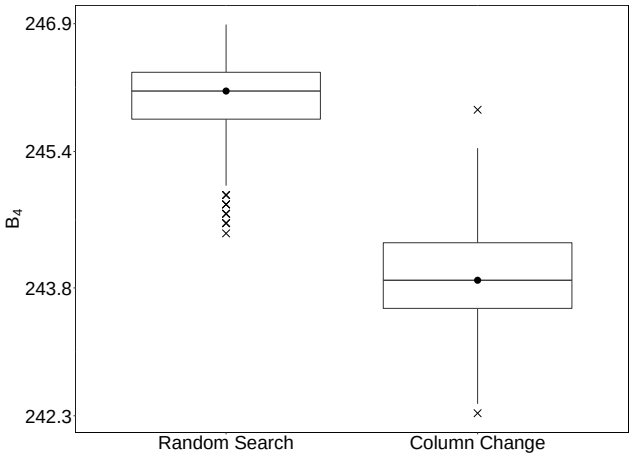


(a) OA64One

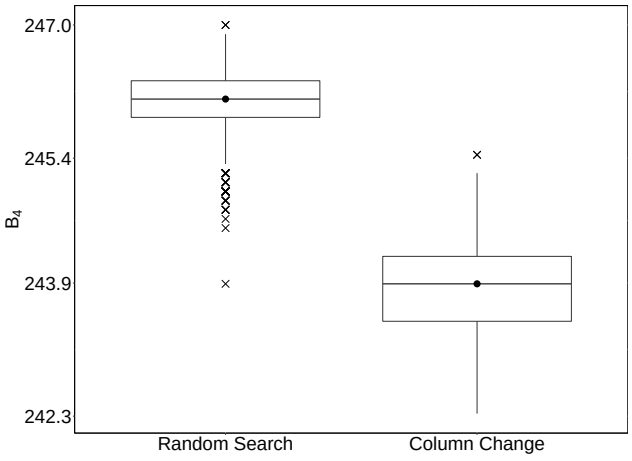
(b) OA64Two



(c) OA80

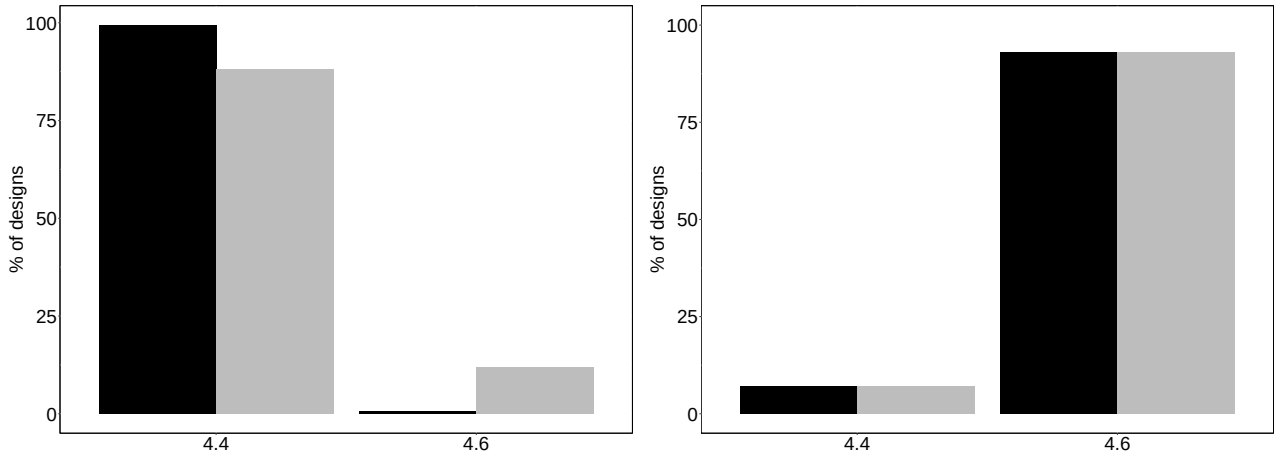


(d) OA96One



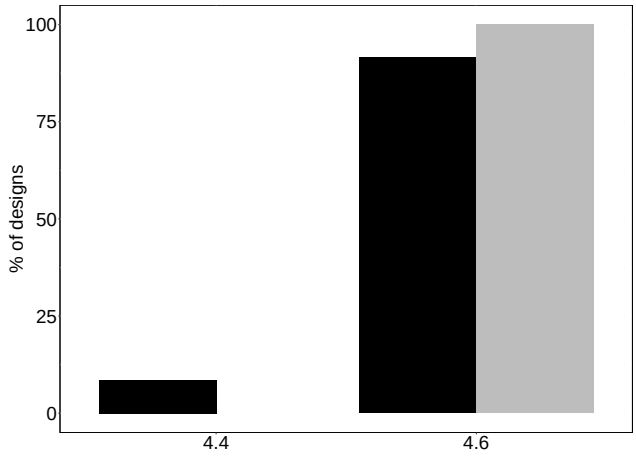
(e) OA96Two

Figure A1: 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.

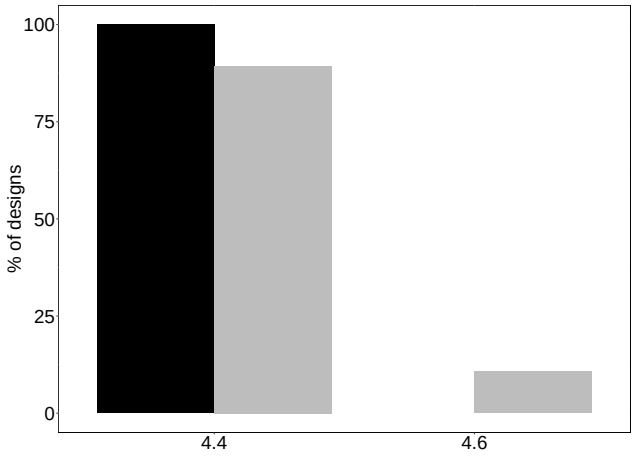


(a) OA64One

(b) OA64Two

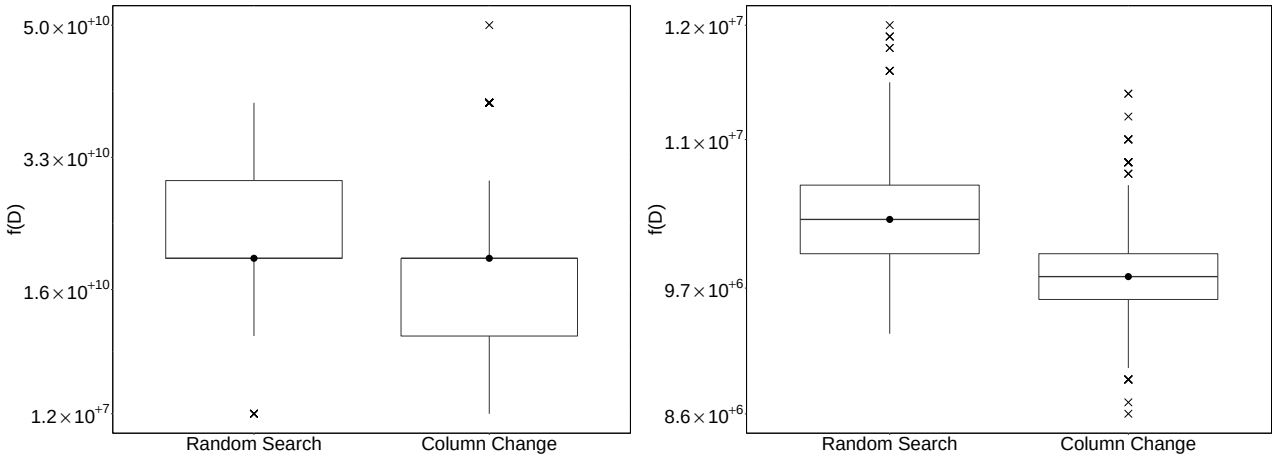


(c) OA80



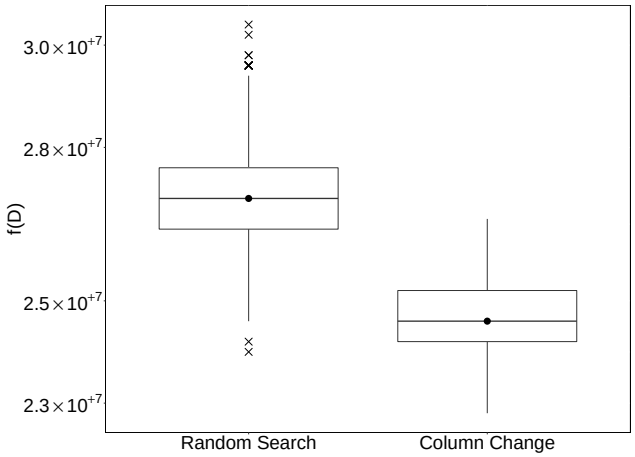
(d) OA96One

Figure A2: Generalized resolution for concatenated designs resulting from random search (black) and the column change algorithm (gray) for four of the design cases of Table A1. 100% corresponds to 1,000 optimized designs.

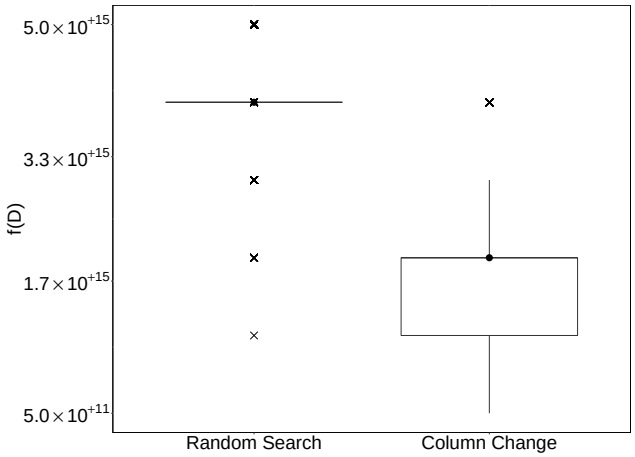


(a) OA64One

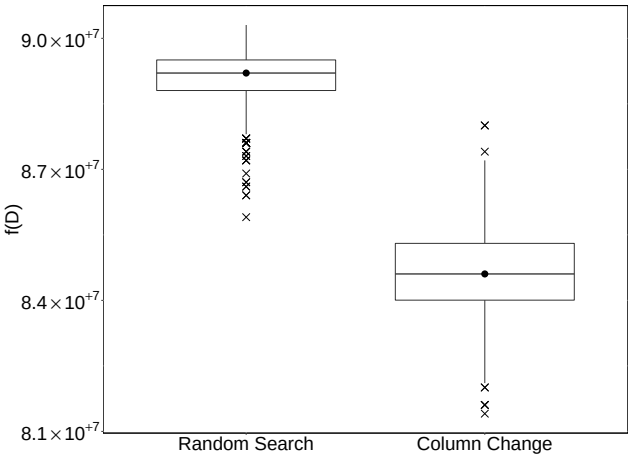
(b) OA64Two



(c) OA80



(d) OA96One



(e) OA96Two

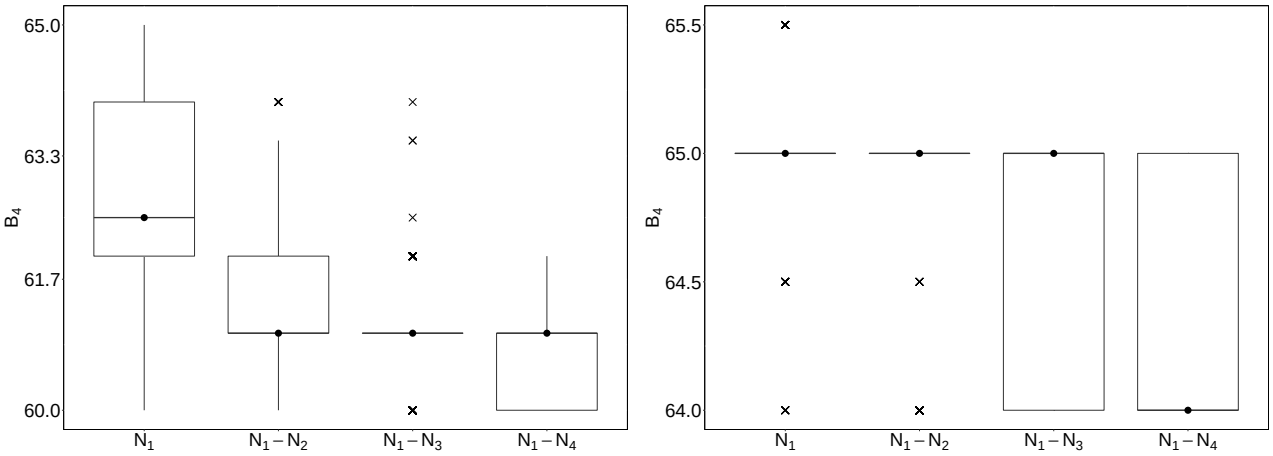
Figure A3: Performance of the column change algorithm and a random search strategy in terms of minimizing the $f(D)$ values. Figures (a), (d) and (e) show $f(D)$ values for all designs resulting from 1,000 starts. In Figures (b) and (c), generated designs with a generalized resolution of 4.25 (OA64Two) or 4.4 (OA80) are disregarded.

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 one, then including neighborhood one and two, and so on. For the 64-run cases, the 80-run cases, 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.

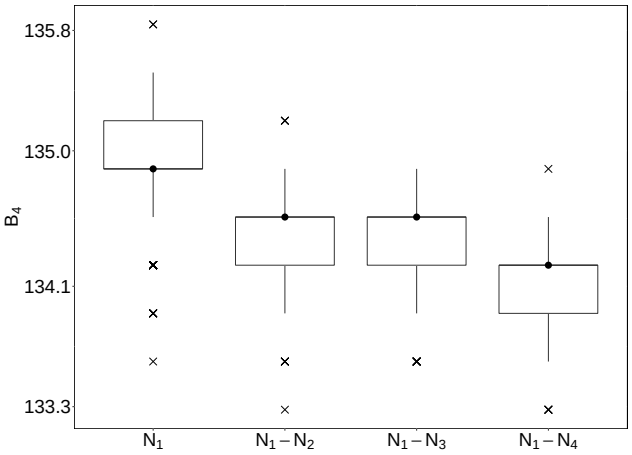
Figure A4 shows box plots of B_4 values of the concatenated designs when adding one neighborhood at a time. For the OA64One case, Figure A4a shows a decrease in the median and the variance of the B_4 values when neighborhoods two and three are introduced successively. Using neighborhoods 1–3, all but 5 concatenated designs have a B_4 value 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 A4b 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 larger number of concatenated designs with B_4 values lower than 65. Figure A4c shows a decrease in the B_4 values of the concatenated designs for instance OA80 when the second and the fourth neighborhood are added; there seems to be no effect of the third neighborhood in this case. Figures A4d and A4e clearly demonstrate the 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 successive inclusion of the four neighborhoods on $f(D)$ is shown in Figure A5. 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 equalling 4.5, 4.6 and 4.67, respectively. The figure shows that a successive inclusion of the neighborhoods one to four generally decreases the objective function value that can be reached by the algorithm. The decrease due to the third neighborhood, however, is substantial only in the OA96One case. For that case, the median over 500 designs decreases from 5×10^{11} to 4.8×10^{11}. As neighborhood 3 is beneficial in at least one case, we retain this neighborhood in the CC/VNS algorithm.

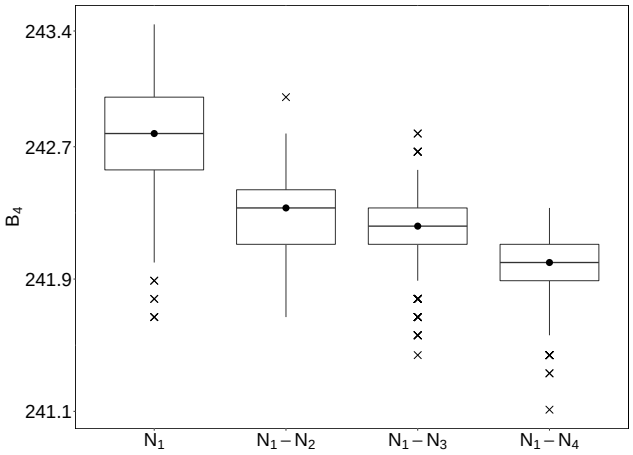


(a) OA64One

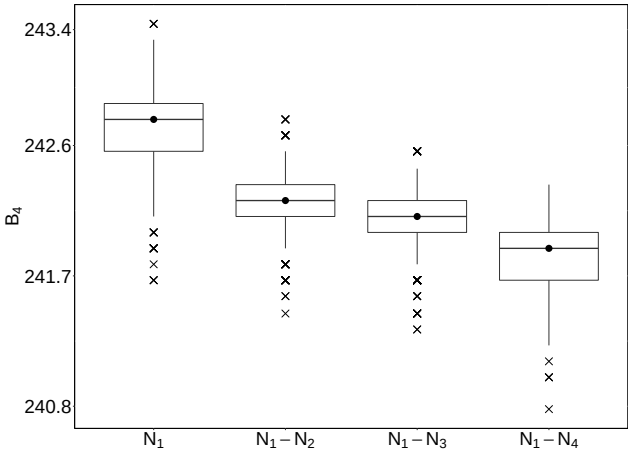
(b) OA64Two



(c) OA80

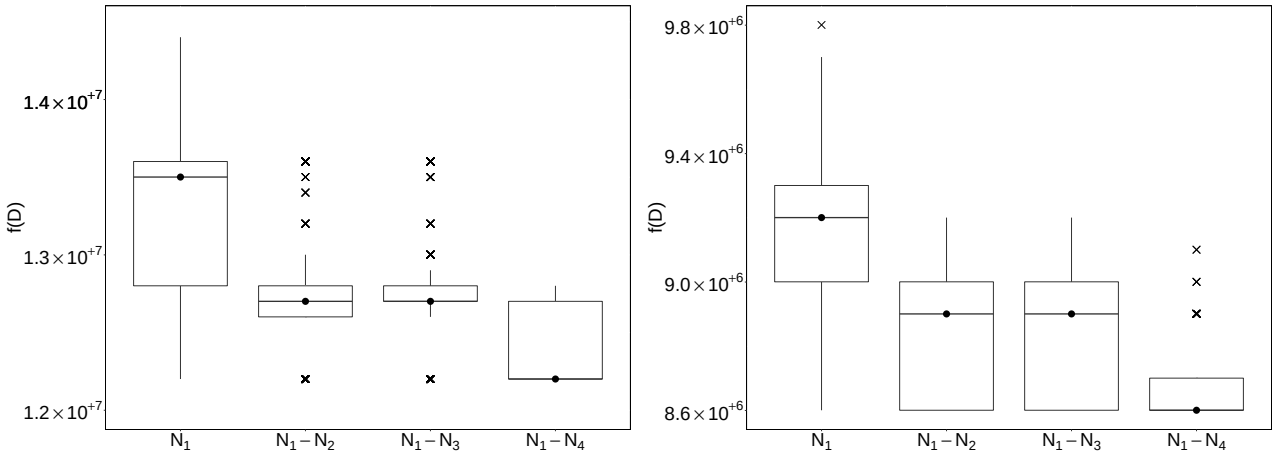


(d) OA96One



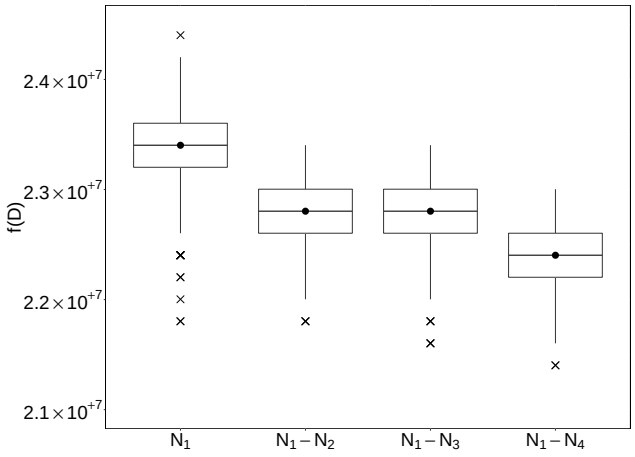
(e) OA96Two

Figure A4: 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 $N_1 - N_4$.

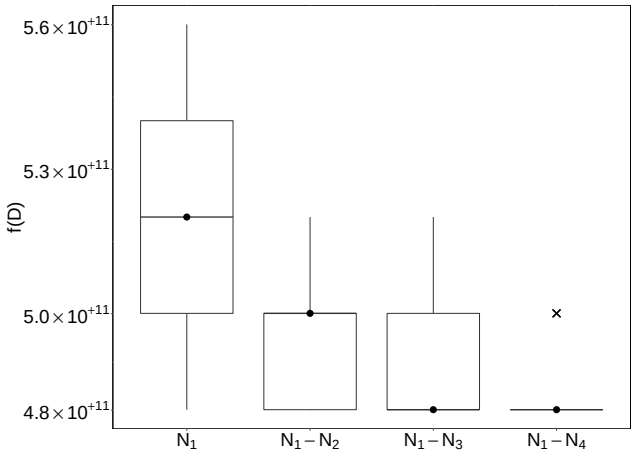


(a) OA64One

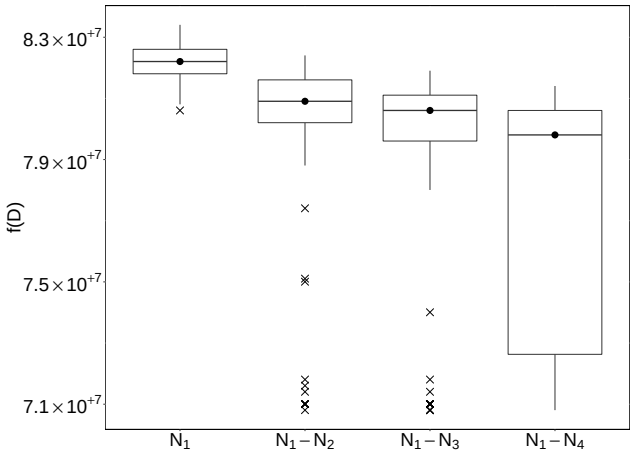
(b) OA64Two



(c) OA80



(d) OA96One



(e) OA96Two

Figure A5: $f(D)$ values for concatenated designs produced by the CC/VNS algorithm using neighborhoods N_1, $N_1 - N_2$, $N_1 - N_3$, or $N_1 - N_4$. Box plots in figures (a), (b), (c), (d) and (e) show results for 470, 500, 500, 93 and 100 designs, respectively.

C Parent designs

We obtained the 32-, 40-, and 48-run parent designs for the 64-, 80-, and 96-run concatenated designs from the complete catalogs available in Schoen et al. (2010). The parent designs we considered for the 64- and 80-run designs that optimize the B_4 value were the OAs with half of these run sizes and that have a minimum B_4 value. The parent designs for 64- and 80-run designs that optimize the F_4 vector were chosen from the top three OAs according to G-aberration with half the run sizes of the concatenated designs. When there are multiple minimum G-aberration designs, we used the rank of the two-factor interaction matrix as tiebreaker. Some series of the arrays of the type $\mathrm{OA}(40, 2^m, 3)$ include more than three minimum G-aberration designs with the maximum rank for the two-factor interaction matrix. We randomly selected three of these designs. Finally, due to the massive number of available 48-run designs (Schoen et al., 2010), we restricted our attention to the designs recommended by Schoen and Mee (2012). The designs of Schoen and Mee (2012) include minimum G- and G_2-aberration designs.

The 56-run strength-3 parent designs for the 112-run concatenated designs were obtained as follows. We started with a complete catalog of 7570 strength-2 OAs with 28 runs and 27 factors (Schoen et al., 2015), which was constructed from the set of all Hadamard matrices of order 28. We then generated all 56-run strength-3 OAs in 27 factors using the fold-over technique. Table A3 shows the nine best designs in terms of the F_4 vector. The first column shows convenient labels for the new 56-run designs. The second column shows the labels of the strength-2 28-run OAs used as '$h.r$', where h corresponds to the Hadamard matrix in Sloane (1999) and r refers to the column used to normalize this matrix. For example, the strength-2 OA 482.4 used to construct 56-run design B is obtained from matrix `had.28.482` in Sloane (1999) by computing the element-wise product of all columns with column 4 and dropping the column of +1s. Finally, columns 3–5 of Table A3 show the F_4 vectors, the B_4 values and the ranks of the two-factor interaction matrices of the new 56-run designs.

Table A3: Best and second best $\mathrm{OA}(56, 2^{27}, 3)$ in terms of the to G-aberration criterion.

Design	Strength-2 OA ($h.r$)	$F_4(40, 24, 8)$	B_4	rank(2FI)
A	487.1	(0, 2106, 15444)	702	27
B	482.4	(6, 2088, 15456)	702	27
C	480.4	(6, 2088, 15456)	702	27
D	377.4	(6, 2088, 15456)	702	27
E	481.1	(6, 2088, 15456)	702	27
F	480.1	(6, 2088, 15456)	702	27
G	377.1	(6, 2088, 15456)	702	27
H	376.1	(6, 2088, 15456)	702	27
I	482.1	(6, 2088, 15456)	702	27

We obtain the 56-run strength-3 parent designs with fewer than 27 factors by projections onto $8 \leq m \leq 26$ factors of the designs in Table A3. For each case, we considered all projections of all nine 56-run OAs with 27 factors. The projections with the smallest G-aberration and the smallest G_2-aberration are characterized in Table A4. The table shows the original 27-factor design from Table A3, the F_4 vector, and the B_4 value. For instance, the best strength-3 56-run design with 10 factors was found by choosing the columns 1, 2, 4, 5, 6, 7, 9, 17, 19, and 21 from design E in Table A3. In all but two cases, the design that minimizes the B_4 value also minimizes the F_4 vector. The exceptions are for 12 and 16 factors, which is why we include two 56-run designs for these numbers of factors.

To construct 112-run concatenated designs with 28 factors, we use the 56-run 27-factor design A as a parent design. For 29-factor designs in 112 runs, we use a parent design constructed by appending the column $\mathbf{z} = [\mathbf{1}_{28}, -\mathbf{1}_{28}]^T$ to the concatenated design produced from design A. Note that, since the complete catalog of 56-run 8-factor designs is available in Schoen et al. (2010), other parent designs might be considered for constructing 112-run designs with 8 factors. Because of the large number of designs in this catalog, however, we use the 8-factor parent design in Table A4 for illustrative purposes.

Table A4: Parent designs of 56 runs and m factors. For all the designs, $F_4(56) = 0$ and the rank of the two-factor interaction matrix equals 27.

m	Design Table A3	$F_4(40, 24, 8)$	B_4	Columns
8	B	(0, 2, 68)	1.76	4 5 6 10 12 20 25 26
9	A	(0, 9, 117)	4.04	1 2 3 4 5 6 17 18 22
10	E	(0, 17, 8)	7.06	1 2 4 5 6 7 9 17 19 21
11	B	(0, 30, 300)	11.63	1 2 4 6 7 10 16 18 20 21 24
12	B	(3, 36, 456)	14.45	1 2 3 4 6 9 10 17 20 21 22 27
12	H	(0, 48, 447)	17.94	1 2 3 4 10 15 18 22 24 25 26 27
13	B	(0, 75, 640)	26.84	1 2 4 7 10 11 12 14 16 22 23 24 25
14	E	(0, 108, 893)	30.6	1 2 3 4 5 6 7 9 16 17 18 19 21 23
15	B	(0, 153, 1212)	52.84	1 2 4 7 9 10 11 12 14 16 17 22 23 24 25
16	B	(4, 192, 1624)	70.45	1 2 4 7 9 10 11 12 14 16 17 22 23 24 25
16	B	(0, 208, 1612)	71.1	1 2 4 7 9 10 11 12 13 14 16 17 22 23 24 25
17	C	(0, 276, 2104)	93.63	1 2 3 5 6 7 8 10 12 13 14 15 18 19 23 24 25
18	C	(0, 359, 2701)	121.6	1 2 3 4 5 8 10 11 13 16 17 18 21 22 23 24 25 26
19	A	(0, 459, 3417)	154.04	1 2 3 4 5 8 10 11 13 16 17 18 21 22 23 24 25 26
20	E	(0, 576, 4269)	192.92	1 2 3 4 5 6 7 8 9 10 11 14 15 17 21 22 23 24 25 27
21	D	(0, 714, 5271)	238.71	1 2 3 4 5 6 7 9 11 12 14 16 17 18 19 21 22 23 24 25 26
22	A	(0, 876, 6439)	292.3	1 2 3 4 5 6 7 9 11 12 14 16 17 18 19 21 22 23 24 25 26
23	A	(0, 1062, 7793)	354.1	$1 - 23$
24	A	(0, 1275, 9351)	425.02	$1 - 24$
25	A	(0, 1518, 11132)	506	$3 - 27$
26	A	(0, 1794, 13156)	598	$2 - 27$
27	A	(0, 2106, 15444)	702	$1 - 27$
28	A	(0, 2457, 18018)	819	$\mathbf{z}, 1 - 27$

The 64-run designs to construct 128-run designs that optimize the B_4 value include the regular minimum aberration designs (Chen et al., 1993), the designs constructed from quaternary linear codes (Xu and Wong, 2007), and our best concatenated designs in terms of the B_4 value. We label these designs 'ma.l', 'xw.l', and 'coa.l', respectively, where l denotes the label of the designs 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 projections. The projections with the best F_4 vectors were used as parent designs. Table A5 shows the best F_4 vectors for $9 \leq m \leq 32$ and the m columns from the folded over Paley matrix to obtain these vectors. The rank of the two-factor interaction matrix 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.

Table A5: Projections from the folded-over Paley Hadamard matrix of order 32 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	$1-32$

D Tables of concatenated designs

We present concatenated designs with 64–128 runs in Tables A6−A11. 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 extra factor $\mathbf{z} = \left[\mathbf{1}_{N/2}, -\mathbf{1}_{N/2}\right]^\top$, 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 rank of the two-factor interaction matrix by $m - 1$; all interactions involving this factor are clear. There is one table for each run size up to 112, and there are separate tables for 128-run designs that optimize the B_4 value and the F_4 vector. The tables report the generalized resolution (R), the F_4 vector, the B_4 value, the rank of the two-factor interaction matrix, and the 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 to obtain the final design after the sign

switch, respectively.

The steps required to construct the concatenated designs from Tables A6−A11 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 as D_s.
3. Order the columns of D_s according to the order indicated by δ. Denote this design as D_{sp}.
4. Concatenate D_u and D_{sp} to generate the concatenated design D.
5. Append the corresponding column $\mathbf{z}$ to D to obtain a 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 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 ordering the columns of design D_s as follows: 6, 3, 4, 5, 2, 8, 9, 1, 7. That is, column 6 in D_s is the first column of design D_{sp}, column 3 in D_s is the second column of D_{sp}, column 4 in D_s is the third column of D_{sp}, and so on. Next, we concatenate the orthogonal array 27 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 a rank of the two-factor interaction matrix equalling 45. So, the design permits estimation of the two factor interactions along with the main effects.

Table A6: Concatenated designs with 64 runs.

Design	D_u, D_l	R	$F_4(64, 48, 32, 16)$	B_4	rank(2FI)	γ	δ
9.b	23, 32	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, 34	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, 21	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, 3	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	$1, 8, 9, 10, 11$	$11, 13, 12, 1, 8, 7, 9, 16, 10, 14, 5, 15, 3, 2, 4, 6$

Table A7: Concatenated designs with 80 runs. $F_4(80, 64) = (0, 0)$ for all designs.

Design	D_u, D_l	R	$F_4(48, 32, 16)$	B_4	rank(2FI)	γ	δ
9.b	105, 105	4.8	(0, 0, 22)	0.88	36	2, 6	1, 4, 7, 2, 5, 3, 6, 8
9.f	105, 96	4.8	(0, 0, 18)	0.72	36	3, 4, 6, 8	3, 6, 2, 7, 4, 5, 8, 1
10.b	213, 213	4.8	(0, 0, 44)	1.76	45	1, 2, 4, 6, 8, 9	4, 8, 5, 3, 9, 7, 1, 6, 2
10.f	213, 213	4.8	(0, 0, 44)	1.76	45	1, 4, 8	8, 1, 4, 2, 9, 5, 6, 7, 3
11.b	353, 353	4.8	(0, 0, 82)	3.28	55	1, 2, 5, 7, 8, 9	4, 1, 8, 2, 9, 5, 6, 3, 10, 7
11.f	353, 353	4.8	(0, 0, 82)	3.28	55	3, 4, 6, 10	10, 8, 5, 1, 6, 2, 9, 7, 3, 4
12.b	112, 255	4.6	(0, 4, 164)	7.2	49	3, 7	3, 11, 9, 4, 6, 2, 8, 5, 7, 10, 1
12.f	67, 67	4.6	(0, 4, 178)	7.76	49	2, 5, 8, 9	9, 1, 5, 11, 8, 7, 6, 10, 2, 4, 3
13.b	157, 157	4.6	(0, 24, 186)	11.28	50	1, 2, 8, 10, 11	9, 1, 7, 6, 3, 8, 10, 12, 11, 5, 2, 4
13.f	157, 157	4.6	(0, 6, 264)	11.52	50	2, 5, 7, 8, 10, 11, 12	11, 1, 5, 3, 8, 2, 4, 10, 7, 6, 12, 9
14.b	36, 126	4.4	(1, 25, 330)	17.56	51	3, 5, 11, 13	7, 2, 12, 13, 6, 3, 9, 10, 5, 8, 4, 11, 1
14.f	55, 55	4.6	(0, 16, 415)	19.16	51	1, 2, 3, 5, 8, 10	8, 3, 2, 4, 10, 9, 12, 1, 6, 5, 13, 7, 11
15.b	38, 38	4.4	(6, 16, 507)	25	52	2, 3, 4, 5, 7, 8, 12, 13	3, 4, 6, 2, 7, 8, 1, 5, 11, 13, 12, 10, 14, 9
15.f	27, 27	4.6	(0, 28, 563)	27	52	2, 4, 6, 11, 13	4, 2, 11, 14, 13, 10, 9, 5, 1, 8, 3, 6, 7, 12
16.b	15, 36	4.4	(1, 55, 658)	35.48	53	2, 3, 5, 7, 8, 11, 12, 13, 14, 15	9, 3, 4, 12, 2, 5, 6, 11, 8, 7, 14, 10, 13, 1, 15
16.f	32, 36	4.6	(0, 43, 757)	37.16	53	2, 3, 5, 7, 9, 10, 11, 13	7, 4, 13, 1, 8, 6, 9, 14, 15, 3, 2, 10, 5, 11, 12
17.b	1, 21	4.4	(1, 78, 886)	48.28	54	4, 9, 10, 12, 15	16, 2, 10, 1, 14, 3, 12, 7, 13, 11, 8, 6, 15, 4, 9,
17.f	4, 4	4.6	(0, 67, 989)	50.28	54	5, 7, 9, 10, 16	9, 2, 13, 14, 12, 11, 6, 1, 16, 15, 7, 8, 3, 4, 5, 10
18.b	2, 4	4.4	(2, 101, 1178)	64	55	2, 8, 9, 12, 13, 16	5, 10, 17, 16, 12, 4, 7, 11, 6, 2, 3, 13, 9, 14, 8, 15, 1
18.f	1, 3	4.6	(0, 95, 1250)	65.2	55	1, 3, 7, 8, 12, 13, 14, 15, 17	2, 1, 11, 15, 12, 17, 4, 13, 9, 14, 7, 6, 8, 16, 5, 3, 10
19.b	1, 5	4.4	(1, 148, 1481)	83.28	56	1, 4, 6, 8, 9, 11, 12, 13, 16, 17	1, 8, 6, 3, 4, 13, 2, 14, 17, 18, 10, 7, 9, 15, 5, 16, 12, 11
19.f	1, 1	4.6	(0, 130, 1614)	85.36	56	5, 6, 7, 8, 10, 14	9, 1, 12, 6, 8, 10, 4, 17, 5, 14, 11, 7, 3, 15, 2, 13, 16, 18
20.b	1, 3	4.4	(1, 186, 1907)	106.4	57	3, 6, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19	1, 13, 16, 17, 19, 3, 10, 14, 2, 8, 7, 5, 9, 12, 4, 11, 6, 18, 15
20.f	3, 3	4.6	(0, 167, 2064)	109.28	57	3, 8, 9, 12, 13, 15	4, 13, 3, 11, 10, 16, 8, 9, 14, 6, 7, 2, 18, 15, 5, 1, 19, 12, 17
21.b	2, 2	4.4	(4, 242, 2320)	132.96	58	5, 7, 8, 9, 11, 13, 14, 17, 18, 19, 20	10, 8, 19, 1, 17, 14, 18, 2, 3, 12, 16, 15, 7, 9, 20, 13, 11, 6, 4, 5
21.f	3, 3	4.6	(0, 216, 2557)	136.84	58	1, 2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 17, 18	20, 1, 12, 19, 5, 6, 16, 15, 17, 8, 10, 9, 14, 4, 11, 7, 2, 3, 18, 13

Table A8: Concatenated designs with 96 runs. $F_4(96, 80, 64) = (0, 0, 0)$ for all designs except 15.b. For that design $F_4(96, 80, 64) = (1, 0, 0)$.

Design	D_u, D_l	R	$F_4(48, 32, 16)$	B_4	rank(2FI)	γ	δ
9.b	8382, 8382	4.83	(0, 0, 6)	0.17	36	2, 4, 6, 7	2, 1, 6, 8, 7, 3, 5, 4
9.f	8382, 8382	4.83	(0, 0, 6)	0.17	36	1, 3, 5, 6, 7	1, 2, 6, 7, 8, 5, 3, 4
10.b	4-16240, 4-16240	4.66	(0, 6, 0)	0.67	45	1, 3, 4, 5, 6, 7, 8	2, 1, 3, 4, 6, 5, 7, 9, 8
10.f	4-16240, 4-16240	4.83	(0, 0, 24)	0.67	45	4, 5, 6, 7, 9	4, 8, 3, 9, 1, 6, 7, 2, 5
11.b	0-136872, 0-136872	4.66	(0, 10, 16)	1.56	54	3, 6, 7	1, 2, 3, 4, 5, 6, 7, 8, 10, 9
11.f	0-136872, 0-136872	4.83	(0, 0, 70)	1.94	55	2, 6, 8, 9	5, 6, 4, 9, 3, 8, 2, 1, 10, 7
12.b	3-351294, 5-32061	4.66	(0, 18, 32)	2.89	62	2, 5, 8, 10, 11	1, 2, 3, 5, 4, 6, 7, 11, 9, 10, 8
12.f	3-351294, 5-32061	4.66	(0, 1, 126)	3.61	66	4, 5, 7, 8, 9, 10, 11	5, 11, 1, 9, 2, 10, 3, 4, 7, 6, 8
13.b	0-541920, 0-541920	4.66	(0, 14, 144)	5.56	67	3, 4, 6, 7, 8, 9, 11, 12	8, 10, 5, 2, 9, 6, 7, 12, 4, 11, 3, 1
13.f	0-541920, 0-541920	4.66	(0, 2, 222)	6.39	77	1, 2, 3, 5, 6, 9, 11, 12	7, 2, 6, 3, 4, 8, 11, 12, 1, 9, 10, 5
14.b	0-594498, 0-594498	4.66	(0, 21, 216)	8.33	69	1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13	6, 12, 10, 4, 5, 1, 7, 8, 11, 3, 9, 2, 13
14.f	0-594498, 0-594498	4.66	(0, 8, 330)	10.06	80	1, 2, 5, 7, 11, 12	10, 1, 11, 8, 6, 9, 13, 3, 4, 7, 5, 12, 2
15.b	2-70173, 2-70173	4	(0, 108, 0)	13	74	1, 3, 6, 8, 13	1, 2, 3, 4, 10, 8, 12, 7, 5, 11, 6, 9, 13, 14
15.f	2-70173, 2-70173	4.66	(0, 31, 440)	15.67	80	1, 3, 4, 5, 8, 9, 10, 11, 12, 13	10, 11, 9, 14, 13, 12, 5, 6, 4, 7, 8, 2, 1, 3
16.b	4-136404, 4-136404	4.66	(0, 100, 614)	28.17	61	3, 4, 7, 9, 14	10, 15, 3, 5, 8, 14, 12, 4, 9, 11, 1, 2, 13, 6, 7
16.f	3-42200, 4-136404	4.66	(0, 71, 762)	29.06	61	1, 5, 6, 11, 13, 14, 15	11, 12, 15, 14, 1, 7, 13, 6, 3, 5, 2, 10, 4, 8, 9
17.b	3-6427, 3-6427	4.5	(3, 120, 875)	38.39	62	1, 2, 4, 5, 7, 10, 12, 14	5, 15, 12, 11, 10, 6, 16, 1, 3, 8, 4, 14, 9, 7, 2, 13
17.f	4-78065, 4-78065	4.66	(0, 102, 1012)	39.44	62	1, 8, 9, 11, 13, 14, 15, 16	6, 16, 8, 4, 12, 9, 11, 5, 7, 14, 1, 15, 3, 10, 2, 13
18.b	4-31172, 4-31172	4.66	(0, 181, 1118)	51.17	63	2, 5, 6, 7, 8, 9, 10, 11, 13, 15, 16, 17	14, 15, 5, 3, 4, 11, 17, 13, 8, 16, 2, 1, 12, 10, 7, 6, 9
18.f	4-31172, 4-31172	4.66	(0, 144, 1312)	52.44	63	1, 3, 4, 5, 8, 10, 12, 13, 14, 15, 16, 17	7, 10, 1, 11, 9, 5, 2, 13, 14, 8, 12, 15, 4, 16, 17, 6, 3
19.b	4-7024, 4-7024	4.66	(0, 223, 1506)	66.61	64	2, 3, 4, 6, 11, 12, 15, 17	7, 3, 18, 14, 1, 13, 9, 2, 8, 11, 10, 17, 12, 5, 16, 4, 6, 15
19.f	4-7024, 4-7024	4.66	(0, 197, 1676)	68.44	64	3, 5, 9, 10, 11, 13, 14, 16, 18	9, 1, 15, 16, 6, 13, 7, 14, 11, 10, 12, 3, 4, 2, 8, 5, 17, 18
20.b	4-659, 4-659	4.66	(0, 300, 1876)	85.44	65	5, 6, 9, 12, 13, 16, 18, 19	2, 6, 16, 4, 18, 1, 10, 15, 14, 7, 12, 17, 19, 8, 3, 5, 9, 11, 13
20.f	4-659, 4-659	4.66	(0, 260, 2092)	87	65	3, 5, 6, 7, 10, 12, 15, 16, 19	2, 3, 7, 19, 9, 16, 10, 8, 17, 14, 12, 1, 15, 11, 4, 6, 13, 5, 18
21.b	4-43, 4-43	4.66	(0, 397, 2286)	107.611	66	3, 4, 7, 10, 11, 12, 13, 15, 17	16, 12, 10, 20, 4, 8, 11, 9, 13, 3, 18, 15, 2, 14, 5, 6, 7, 19, 17, 1
21.f	4-43, 4-43	4.66	(0, 333, 2624)	109.89	66	1, 2, 3, 7, 11, 13, 14, 15, 16, 17	15, 7, 16, 6, 4, 3, 9, 5, 17, 10, 1, 14, 2, 8, 19, 12, 20, 11, 18, 13
22.b	4-2, 4-2	4.66	(0, 477, 2918)	134.06	67	4, 5, 7, 12, 16, 17, 18, 19	19, 9, 11, 3, 17, 13, 5, 10, 4, 16, 7, 21, 18, 20, 1, 15, 6, 14, 2, 8, 12
22.f	4-2, 4-2	4.66	(0, 424, 3224)	136.67	67	1, 2, 3, 5, 8, 9, 14, 15, 18, 20	2, 15, 12, 20, 13, 4, 16, 17, 3, 9, 1, 19, 7, 14, 8, 5, 18, 10, 11, 6, 21
23.b	4-1, 4-1	4.66	(0, 608, 3500)	164.78	68	1, 3, 4, 5, 6, 7, 9, 11, 13, 14, 16, 17, 18, 20, 22	13, 8, 14, 3, 11, 19, 7, 2, 10, 16, 15, 22, 21, 20, 6, 18, 5, 4, 17, 1, 9, 12
23.f	4-1, 4-1	4.66	(0, 531, 3906)	167.5	68	1, 2, 3, 6, 7, 8, 10, 13, 14, 15, 18, 21, 22	14, 15, 17, 18, 7, 2, 11, 6, 22, 12, 20, 8, 19, 1, 3, 21, 13, 4, 5, 16, 9, 10
24.b	4-1, 4-1	4.66	(0, 695, 4434)	200.39	69	1, 5, 14, 18, 19, 20, 21, 22	3, 6, 20, 4, 1, 21, 7, 23, 19, 15, 14, 16, 5, 11, 13, 2, 22, 9, 10, 12, 8, 18, 17
24.f	4-1, 4-1	4.66	(0, 583, 4996)	203.56	69	1, 4, 5, 7, 9, 10, 14, 15, 16, 17, 18, 20, 21, 2220	23, 10, 3, 15, 13, 11, 6, 18, 12, 22, 9, 4, 16, 2, 21, 14, 5, 8, 20, 17, 7, 19, 1
25.b	60, 60	4.66	(0, 861, 5240)	241.22	70	1, 2, 3, 5, 7, 8, 9, 12, 13, 14, 18, 21, 22, 24	22, 6, 4, 5, 21, 11, 7, 23, 16, 15, 17, 3, 8, 12, 14, 19, 1, 13, 24, 2, 18, 10, 9, 20
25.f	60, 60	4.66	(0, 708, 5984)	244.89	70	1, 4, 5, 8, 10, 12, 13, 14, 16, 18, 22, 24	22, 15, 4, 23, 20, 18, 8, 5, 2, 11, 7, 3, 9, 21, 19, 17, 1, 16, 13, 6, 12, 10, 14, 24

Table A9: Concatenated designs with 112 runs. $F_4(112, 96, 80, 64) = (0, 0, 0, 0)$ for all designs. Designs 13.b, 13.f, and 17.f are constructed from the 56-run 12-factor or 16-factor designs that are best in terms of the F_4 vector (see Table A4). The parent designs of the concatenated design 17.b are the best 56-run 16-factor design in terms of the B_4 value (D_u) and the best 56-run 16-factor design in terms of the F_4 vector (D_l).

Design	R	$F_4(48, 32, 16)$	B_4	rank(2FI)	γ	δ
9.b	4.86	(0, 0, 18)	0.37	36	1, 2, 3, 5	1, 5, 3, 4, 2, 6, 8, 7
9.f	4.86	(0, 0, 20)	0.41	36	1, 3, 6, 7	6, 5, 3, 4, 2, 1, 7, 8
10.b	4.86	(0, 0, 60)	1.23	45	2, 3, 4, 5, 6	7, 1, 6, 9, 3, 2, 5, 4, 8
10.f	4.86	(0, 0, 60)	1.23	45	1, 4, 5, 7, 8	9, 1, 6, 2, 4, 3, 7, 5, 8
11.b	4.57	(1, 0, 101)	2.25	55	1, 2, 3, 4, 5, 9	1, 2, 3, 4, 9, 6, 7, 10, 5, 8
11.f	4.71	(0, 2, 112)	2.45	55	2, 5, 7, 10	8, 2, 6, 4, 5, 3, 7, 1, 9, 10
12.b	4.57	(1, 10, 149)	4.04	65	2, 5, 7, 8, 10	4, 10, 3, 1, 5, 6, 7, 8, 9, 2, 11
12.f	4.71	(0, 10, 174)	4.37	65	1, 3, 4, 6, 7	7, 1, 4, 2, 5, 6, 8, 11, 3, 10, 9
13.b	4.71	(0, 16, 250)	6.41	65	3, 4, 10, 11, 12	10, 4, 6, 12, 11, 1, 9, 2, 5, 3, 7, 8
13.f	4.71	(0, 16, 250)	6.41	65	1, 2, 3, 5, 6, 7, 8, 9, 12	10, 8, 4, 12, 9, 3, 5, 1, 11, 2, 7, 6
14.b	4.57	(1, 49, 334)	11	67	1, 2, 4, 7, 8, 9, 11, 12	3, 4, 11, 13, 6, 5, 12, 1, 8, 2, 10, 9, 7
14.f	4.71	(0, 43, 409)	11.86	67	2, 3, 4, 5, 6, 7, 10, 13	4, 11, 3, 8, 2, 10, 9, 7, 1, 12, 5, 6, 13
15.b	4.57	(3, 69, 490)	16.18	68	1, 4, 6, 9, 11, 12	2, 10, 11, 4, 7, 8, 14, 9, 5, 13, 1, 3, 6, 12
15.f	4.71	(0, 72, 531)	16.71	68	2, 4, 5, 10, 11, 12, 14	9, 13, 1, 6, 3, 7, 11, 12, 10, 8, 5, 14, 2, 4
16.b	4.57	(2, 114, 639)	22.71	69	1, 2, 3, 4, 8, 9, 11, 12, 13, 15	12, 15, 9, 11, 14, 7, 3, 5, 6, 8, 4, 10, 13, 2, 1
16.f	4.71	(0, 112, 725)	23.94	69	2, 5, 6, 7, 11, 12, 14, 15	12, 3, 10, 7, 2, 4, 9, 13, 15, 14, 5, 8, 6, 11, 1
17.b	4.57	(6, 155, 866)	31.43	70	3, 4, 6, 8, 9, 13	6, 7, 10, 9, 4, 2, 11, 15, 1, 16, 5, 8, 3, 12, 13, 14
17.f	4.71	(0, 161, 981)	33.16	70	2, 3, 6, 11, 12, 15, 16	6, 1, 7, 3, 8, 14, 11, 13, 16, 2, 4, 12, 5, 9, 15, 10
18.b	4.57	(16, 197, 1138)	42.25	71	1, 2, 3, 4, 7, 10, 12, 13	1, 7, 9, 4, 13, 3, 8, 15, 11, 5, 2, 6, 16, 12, 17, 10, 14
18.f	4.71	(0, 232, 1216)	43.76	71	3, 5, 7, 11, 12, 14	11, 3, 2, 14, 15, 12, 17, 9, 4, 16, 1, 8, 5, 13, 10, 6, 7
19.b	4.57	(15, 261, 1519)	55.06	72	1, 4, 5, 9, 12, 16	7, 3, 16, 11, 6, 13, 14, 9, 12, 5, 4, 2, 18, 15, 1, 8, 17, 10
19.f	4.71	(0, 314, 1594)	58.16	72	1, 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 18	18, 6, 15, 13, 3, 9, 7, 1, 8, 5, 11, 14, 10, 4, 12, 16, 17, 2
20.b	4.57	(25, 331, 1929)	70.98	73	5, 7, 8, 9, 11, 15, 17	1, 12, 2, 18, 9, 16, 11, 13, 15, 19, 14, 7, 17, 10, 5, 3, 4, 8, 6
20.f	4.71	(0, 425, 1912)	73.71	73	1, 2, 6, 7, 9, 11, 12, 16, 18	2, 8, 3, 16, 11, 9, 14, 1, 12, 5, 4, 10, 6, 19, 18, 13, 15, 7, 17
21.b	4.57	(21, 447, 2417)	89.67	74	2, 3, 5, 6, 7, 10, 12, 13, 14, 17, 19	17, 19, 1, 10, 15, 9, 16, 2, 4, 18, 12, 8, 14, 7, 20, 11, 13, 3, 5, 6
21.f	4.57	(1, 559, 2383)	94.45	74	1, 2, 3, 4, 5, 6, 8, 10, 11, 15, 17, 18, 19	14, 17, 12, 5, 8, 7, 1, 20, 4, 9, 16, 10, 15, 19, 3, 2, 13, 18, 6, 11
22.b	4.57	(25, 578, 2930)	111.57	75	1, 3, 4, 5, 11, 12, 16, 17, 18, 19, 20, 21	1, 6, 5, 2, 4, 13, 17, 10, 19, 16, 3, 7, 12, 18, 21, 8, 11, 15, 14, 20, 9
22.f	4.57	(3, 674, 3040)	117.61	75	1, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16	13, 6, 21, 10, 12, 15, 2, 18, 9, 7, 11, 19, 4, 16, 3, 1, 5, 20, 17, 8, 14
23.b	4.57	(41, 716, 3506)	137.53	76	2, 3, 8, 13, 15, 16, 17, 22	3, 1, 6, 17, 10, 14, 20, 15, 18, 16, 5, 22, 8, 19, 9, 21, 7, 2, 4, 12, 13, 11
23.f	4.57	(8, 805, 3661)	141.89	76	2, 3, 9, 10, 12, 13, 14, 18, 19, 21, 22	5, 4, 21, 6, 14, 9, 19, 10, 22, 17, 11, 12, 7, 3, 1, 13, 18, 8, 20, 15, 2, 16
24.b	4.57	(58, 848, 4301)	167.65	77	3, 6, 8, 13, 14, 18, 21, 22, 23	14, 11, 1, 12, 3, 22, 2, 7, 15, 4, 20, 8, 21, 10, 18, 6, 13, 5, 23, 9, 19, 16, 17
24.f	4.57	(13, 1000, 4452)	174.88	77	2, 3, 8, 10, 12, 13, 15, 20, 21, 22, 23	11, 2, 5, 18, 8, 15, 10, 14, 19, 21, 23, 12, 20, 16, 3, 17, 6, 4, 1, 7, 9, 13, 22
25.b	4.57	(59, 1008, 5341)	202.12	78	1, 2, 4, 5, 6, 8, 12, 13, 14, 17, 20, 21, 22, 23	7, 10, 8, 12, 9, 21, 11, 13, 1, 20, 2, 6, 16, 4, 14, 22, 17, 18, 3, 15, 19, 24, 5, 23
25.f	4.57	(17, 1185, 5246)	206.92	78	1, 2, 3, 4, 6, 12, 14, 18, 23	19, 17, 1, 24, 8, 5, 12, 11, 13, 10, 21, 9, 6, 4, 7, 14, 23, 22, 16, 2, 3, 20, 18, 15
26.b	4.57	(79, 1225, 6219)	241.43	79	1, 2, 3, 4, 5, 9, 11, 12, 13, 16, 19, 21, 22, 25	1, 6, 3, 14, 25, 23, 7, 8, 5, 19, 12, 17, 9, 22, 20, 2, 10, 16, 18, 15, 24, 11, 13, 4, 21
26.f	4.57	(24, 1408, 6312)	248.16	79	2, 4, 5, 6, 7, 9, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24	23, 17, 19, 13, 8, 3, 1, 2, 14, 20, 7, 15, 6, 5, 25, 12, 16, 24, 22, 21, 11, 10, 9, 18, 4
27.b	4.57	(89, 1460, 7393)	286.41	80	1, 3, 4, 6, 8, 9, 10, 13, 14, 15, 17, 18, 23, 25	21, 5, 7, 11, 1, 25, 4, 3, 8, 14, 17, 24, 10, 9, 19, 12, 26, 20, 16, 22, 23, 15, 6, 2, 18, 13
27.f	4.57	(31, 1674, 7369)	292.74	80	1, 4, 7, 9, 10, 12, 14, 16, 17, 18, 19, 22, 23	13, 19, 4, 17, 9, 18, 7, 3, 2, 5, 16, 23, 11, 20, 26, 22, 21, 15, 10, 24, 6, 1, 25, 14, 8, 12
28.b	4.57	(90, 1760, 8642)	336.57	81	1, 4, 8, 10, 13, 16, 17, 18, 19, 21, 23, 24, 25, 26, 27	15, 27, 24, 7, 20, 19, 21, 13, 3, 26, 16, 17, 9, 10, 6, 5, 11, 22, 12, 2, 4, 1, 23, 8, 14, 25, 18
28.f	4.57	(40, 1987, 8658)	346.25	81	1, 2, 3, 4, 11, 16, 17, 21, 23, 24, 26	18, 21, 6, 15, 9, 5, 13, 17, 10, 2, 22, 20, 26, 23, 19, 24, 8, 4, 14, 11, 27, 7, 16, 25, 12, 3, 1
29.b	4.57	(113, 2066, 10037)	394.25	82	7, 9, 10, 12, 13, 14, 17, 18, 19, 20, 23, 24, 25	19, 13, 25, 3, 23, 2, 20, 12, 14, 17, 24, 15, 5, 7, 21, 18, 22, 8, 10, 9, 27, 16, 28, 6, 4, 11, 1, 26
29.f	4.57	(52, 2296, 10175)	404.63	82	2, 5, 8, 12, 15, 16, 17, 23, 24, 27	5, 14, 19, 23, 28, 10, 27, 1, 26, 4, 21, 25, 6, 15, 22, 7, 12, 9, 20, 2, 17, 11, 13, 8, 3, 18, 24, 16

Table A10: 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	rank(2FI)	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	rank(2FI)	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, P41	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 A11: 128-run concatenated designs that optimize the B_4 value.

Design	D_u, D_l	R	$F_5(96, 64, 32)$	B_5	rank(2FI)	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	rank(2FI)	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.27-32.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	$3, 4, 6, 8, 14, 17, 18, 21, 24, 28, 30, 32$	$23, 5, 16, 3, 20, 22, 8, 18, 30, 2, 27, 32, 6, 12, 24, 21, 7, 14, 15, 25, 4, 17, 19, 29, 26, 1, 11, 10, 13, 28, 9, 31$

Acknowledgments

The research that led to this paper was financially supported by the Flemish Fund for Scientific Research FWO.

References

Avanthay, C., Hertz, A., and Zufferey, N. (2003). A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151:379–388.

Block, R. M. and Mee, R. W. (2005). Resolution IV designs with 128 runs. *Journal of Quality Technology*, 37:282–293.

Bulutoglu, D. A. and Ryan, K. J. (2015). Algorithms for finding generalized minimum aberration designs. *Journal of Complexity*, 31(4):577 – 589.

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

Butler, N. A. (2004). Minimum G_2-aberration properties of two-level foldover designs. *Statistics & Probability Letters*, 67:121–132.

Butler, N. A. (2007). Results for two-level fractional factorial designs of resolution IV or more. *Journal of Statististical Planning and Inference*, 137:317–323.

Caporossi, G. and Hansen, P. (2004). Variable neighborhood search for extremal graphs. 5. Three ways to automate finding conjectures. *Discrete Mathematics*, 276:81–94.

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.

Cheng, C.-S., Mee, R. W., and Yee, O. (2008). Second order saturated orthogonal arrays of strength three. *Statistica Sinica*, 18:105–119.

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

Draguljić, D., Woods, D. C., Dean, A. M., Lewis, S. M., and Vine, A.-J. E. (2014). Screening strategies in the presence of interactions. *Technometrics*, 56:1–1.

Eglese, R. (1990). Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46:271–281.

Fleszar, K. and Hindi, K. S. (2004). Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155:402–413.

Garroi, J.-J., Goos, P., and Sörensen, K. (2009). A variable-neighbourhood search algorithm for finding optimal run orders in the presence of serial correlation. *Journal of Statistical Planning and Inference*, 139:30–44.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467.

Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2008). Variable neighbourhood search: methods and applications. *4OR*, 6:319–360.

Hedayat, A., Sloane, N., and Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications*. Springer Series in Statistics. Springer New York.

Kytöjoki, J., Nuortio, T., Bräysy, O., and Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers and Operations Research*, 34:2743–2757.

Li, W. and Lin, D. K. J. (2003). Optimal foldover plans for two-level fractional factorial designs. *Technometrics*, 45:142–149.

Li, W. and Lin, D. K. J. (2016). A note on foldover of 2^{k-p} designs with column permutations. *Technometrics*, 58:508–512.

Li, W., Lin, D. K. J., and Ye, K. Q. (2003). Optimal foldover plans for two-level nonregular orthogonal designs. *Technometrics*, 45:347–351.

Ma, C. X. and Fang, K. T. (2001). A note on generalized aberration in factorial designs. *Metrika*, 53:85–93.

Michalewicz, Z. and Fogel, D. (2004). *How to Solve It: Modern Heuristics*. Springer.

Miller, A. and Sitter, R. R. (2001). Using the folded-over 12-run Plackett-Burman design to consider interactions. *Technometrics*, 43:44–55.

Mladenović, N., Petrović, J., Kovačević-Vujčić, V., and Čangalović, M. (2003). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, 151:389–399.

Sartono, B., Goos, P., and Schoen, E. (2015). Constructing general orthogonal fractional factorial split-plot designs. *Technometrics*, 57:488–502.

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.

Schoen, E. D. and Mee, R. W. (2012). Two-level designs of strength 3 and up to 48 runs. *Journal of the Royal Statistical Society Series C*, 61:163–174.

Schoen, E. D., Vo-Thanh, N., and Goos, P. (2015). Two-level orthogonal designs in 24 and 28 runs. Technical report, University of Antwerp, Faculty of Applied Economics.

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

Syafitri, U. D., Sartono, B., and Goos, P. (2015). I-optimal design of mixture experiments in the presence of ingredient availability constraints. *Journal of Quality Technology*, 47:220–234.

Tang, B. and Deng, L.-Y. (1999). Minimum G_2-aberration for nonregular fractional factorial designs. *The Annals of Statistics*, 27:1914–1926.

Xu, H. (2005). Some nonregular designs from the Nordstrom-Robinson code and their statistical properties. *Biometrika*, 92:385–397.

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.

Xu, H. and Wu, C. F. J. (2001). Generalized minimum aberration for asymmetrical fractional factorial designs. *The Annals of Statistics*, 29:1066–1077.