TECHNOMETRICS
2019, VOL. 61, NO. 2, 219–232
https://doi.org/10.1080/00401706.2018.1473798

Taylor & Francis
Taylor & Francis Group

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

Alan R. Vazquez[a,b], Peter Goos[a,b], and Eric D. Schoen[b,c]

[a]Department of Engineering Management, University of Antwerp, Antwerp, Belgium; [b]Department of Biosystems, KU Leuven, Leuven, Belgium; [c]TNO, Zeist, The Netherlands

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 they consist of $N/2$ mirror image pairs, these designs are fold-over designs. They are called even and provide at most $N/2-1$ degrees of freedom to estimate interactions. For $k < N/3$ factors, there exist strength-3 designs that are not fold-over designs. They are called even-odd designs and they provide many more degrees of freedom to estimate interactions. 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. 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 and 128 runs involving up to 33 factors, because this allows a comparison with benchmark designs from the literature. With a few exceptions, our even-odd designs outperform or are competitive with the benchmark designs in terms of the aliasing of two-factor interactions and in terms of the available degrees of freedom to estimate two-factor interactions. Supplementary materials for the article are available online.

ARTICLE HISTORY
Received October 2016
Revised April 2018

KEYWORDS
Even-odd design; Generalized aberration; Local search; Second-order saturated; Two-factor interaction; Variable neighborhood search

1. Introduction

Two-level screening designs can be used as experimental plans to identify, from a list of potentially influential factors, those that are indeed influential; see Mee, Schoen, and Edwards (2017) for a recent review of these designs. Many two-level screening designs currently in use involve orthogonal arrays (OAs). Denoting the two levels for each factor by -1 and $+1$, these OAs have main effect contrast vectors that are level-balanced and orthogonal to each other.

This article is motivated by two practical experiments involving many runs and factors that were conducted using an OA. The first one is an enzyme stability experiment conducted at TNO, Zeist, The Netherlands. To improve the stability of an enzyme in a watery solution at room temperature, 17 possible additives were considered. The experiment actually carried out had 64 runs. The second experiment involves a sensitivity analysis of a simulation model for a software process (Houston et al. 2001), and used a design with 30 factors and 64 runs. In this article, we propose a method to construct high-quality designs for these two experiments. The method is suitable for constructing large two-level OAs involving many runs and factors in general. We revisit the enzyme stability experiment and the software process simulation experiment after introducing and evaluating our method.

OAs of strength t are such that all 2^t level combinations of any set of t factors occur equally often (Hedayat, Sloane, and Stufken 1999). Consequently, an OA of strength t has a run size that is a multiple of 2^t. A strength of 2 implies that main effect (ME) contrast vectors are orthogonal to each other but not to two-factor interaction (2FI) contrast vectors. This feature may make it hard to find out whether it is the main effect of one of the factors or the interaction of two other factors that causes a change in the responses. On the positive side, the run sizes of strength-2 OAs can be small. For example, the smallest 17-factor strength-2 OA involves 20 runs; a 20-run 17-factor design that minimizes the aliasing between the MEs and the 2FIs can be found in Sun, Li, and Ye (2008). The smallest 30-factor strength-2 OA involves 32 runs. Textbooks such as Mee (2009) and Wu and Hamada (2009) provide suitable design options for this case.

OAs of strength 3 allow the main effects to be estimated independently from the 2FIs. They are attractive options whenever several 2FIs are suspected to be active, but they have larger run sizes. For 17 and 30 factors, the smallest strength-3 OAs have 40 and 64 runs, respectively. In general, for an N-run k-factor OA of strength 3, $N \geq 2k$. Butler (2004, 2007) showed that all strength-3 OAs for which $k \geq N/3$ must be even designs, which are also called fold-over designs (Cheng, Mee, and Yee 2008). In these designs, half of the runs are mirror images of the other half, in the sense that the signs of the factor levels are switched. A weakness of even designs is that they provide at most $N/2-1$ degrees of freedom for estimating 2FIs. For $k < N/3$ factors, there may be strength-3 designs that cannot be constructed by folding over. These designs, called even-odd designs, generally provide many more degrees of freedom for estimating 2FIs. Therefore, they

CONTACT Alan R. Vazquez alan.vazquezalcocer@kuleuven.be
Supplementary materials for this article are available online. Please go to http://www.tandfonline.com/r/TECH.

are attractive to experimenters who want to estimate a substantial number of interactions along with the main effects. Because their run size N should be larger than $3k$, even-odd designs must have at least 56 and 96 runs for experiments with 17 and 30 factors, respectively.

Complete catalogs exist for two-level strength-3 OAs with up to 48 runs (Schoen, Eendebak, and Nguyen 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 even-odd k-factor 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 level equals $+1$ for the first $N/2$ runs and -1 for the last $N/2$ runs can be added to the concatenated design. Strength-3 designs constructed in this way have run sizes that are multiples of 16. This approach can thus be used to construct even-odd 64-run and 128-run designs based on existing strength-3 designs with 32 and 64 runs, respectively.

Several authors have constructed even-odd designs using variants of the above general approach. Li and Lin (2003), Li, Lin, and Ye (2003), and Cheng, Mee, and Yee (2008) concatenated two copies of a single parent design and subsequently switched 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, Lin, and Ye (2003), Cheng, Mee, and Yee (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 by permuting rows or columns, and switching the signs of the elements in one or more columns). There are two problems with these approaches. 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 sets of columns in which to switch the signs and 3.55687×10^{14} possible column permutations.

The first contribution of this article is to develop an efficient algorithmic procedure to construct even-odd designs by concatenating two strength-3 parent designs, which may or may not be isomorphic. 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 article is to generate new two-level concatenated designs with 64 and 128 runs and up to 33 factors, for instance, for the 17-factor enzyme stability experiment and the 30-factor software process simulation experiment. We compare the newly generated designs with the best designs from the literature. In addition to the benchmark designs of Li and Lin (2016), our comparisons involve the regular 64-run designs reported in Chen, Sun, and Wu (1993), the 17-factor 64-run design from Cheng, Mee, and Yee (2008), the regular 128-run designs reported in Block and Mee (2005) and Xu (2009), the 64- and 128-run designs constructed by Xu and Wong (2007) using quaternary linear codes, and the 64-run designs generated from projections of the folded-over 32-run Hadamard matrix given by the Paley construction. To our knowledge, our study of the projections from this folded-over 32-run Hadamard matrix is new to the literature. Most of 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 degrees of freedom they provide to estimate two-factor interactions.

The rest of this article 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 compares the new designs with 64 and 128 runs with the benchmark designs from the literature. We return to the motivating examples in Section 5, and conclude with a discussion and some suggestions for future research in Section 6.

2. Classification of Strength-3 Designs

Orthogonal two-level designs of strength 3 are most commonly evaluated in terms of their G-aberration and generalized resolution (Deng and Tang 1999), their G_2-aberration (Tang and Deng 1999), and the rank of the matrix consisting of the 2FI contrast vectors (Cheng, Mee, and Yee 2008). We briefly review all of these criteria.

All but the last of the criteria are 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 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 nonnegative integer. A four-factor interaction contrast vector can be calculated as the product of two two-factor interaction contrast vectors. Therefore, whenever a J_4-characteristic of N occurs in an N-run design, this implies that three pairs of two-factor interactions are completely aliased. Whenever J_4-characteristics of zero occur in a design, 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 maximum J_4 value for a given strength-3 design determines its generalized resolution. More specifically, the generalized resolution equals $5-\max(J_4)/N$ for a strength-3 design. So, a large maximum J_4-characteristic implies a small generalized resolution. Ideally, the generalized resolution of a design is large.

The frequencies of the J_s-characteristics calculated for all s-factor interaction contrast vectors are generally collected in a vector F_s. For strength-3 designs, the entries of the F_4 vector are the frequencies of the J_4-characteristic values N, $N-16$, $N-32$, etc. The frequency of the zero value is usually omitted. The concatenated vector $(F_4, F_5, F_6, \ldots, F_k)$ is the confounding frequency vector (CFV) of a strength-3 k-factor design. Ideally, the leftmost elements of the CFV are small, because this

means that there is little severe aliasing between the low-order interactions.

To determine the G-aberration of a k-factor design, all available designs are sorted according to the entries of the CFV $(F_4, F_5, F_6, \ldots, F_k)$, from left to right. The G-aberration of the design is its ranking after the sorting procedure, and a minimum G-aberration design has the best of the rankings. In this article, we restrict our attention to the F_4 vector, because this vector quantifies the most serious aliasing in strength-3 designs, namely, the aliasing among the 2FIs.

Like the G-aberration, the G_2-aberration is determined by sorting all available designs according to a vector. The vector used for the G_2-aberration is called the generalized word length pattern (GWLP), $(B_1, B_2, \ldots, B_k)$, where B_i is the sum of the squared J_i-characteristics divided by N^2. A minimum G_2-aberration design has the best of the rankings after sorting all available designs according to the entries of the GWLP from left to right. For strength-3 OAs, $B_1 = B_2 = B_3 = 0$, and $B_4 > 0$. The B_i values are called generalized word counts. The most important of these is B_4, because it quantifies the aliasing among the 2FIs. Ideally, it is small.

Finally, strength-3 OAs permit the estimation of the intercept and all the main effects simultaneously. The rank of the 2FI contrast matrix quantifies the number of degrees of freedom available for estimating two-factor interactions (Cheng, Mee, and Yee 2008). In the rest of this article, we use the term "degrees of freedom for two-factor interactions" to refer to this criterion.

3. Algorithmic Construction of Even-Odd Designs

In this section, we first describe the principles to concatenate two orthogonal arrays of strength 3. Next, 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 algorithm making small structured changes to one of the parent designs. The CC algorithm is embedded in a variable neighborhood search (VNS) algorithm that investigates increasingly diverse new versions of that parent design. The complete algorithm, called the CC/VNS algorithm, reduces the number of evaluations needed to test column permutations and subsets of columns in which to switch the signs. We conclude this section with an evaluation of the CC/VNS algorithm and with a study concerning the choice of parent designs.

3.1 Concatenation Principles

Our CC/VNS 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. We call the added column the indicator factor, because it identifies the two original OAs. The m 2FIs involving this factor are all orthogonal to the 2FIs of the original factors. As a result, the B_4 value and the F_4 vector of a concatenated design are not affected by adding the indicator factor. Since a strength-3 orthogonal array with $N/2$ runs can accommodate at most $N/4$ factors, the concatenated design has N runs and involves at most $k = N/4 + 1$ factors, including the indicator factor. In this respect, our approach is similar to that of Cheng, Mee, and Yee (2008). We refer to D_u, D_l, and D as the upper design, the lower design, and the concatenated design, respectively. We call D_u and D_l parent designs.

The first step in the construction process is the selection of two strength-3 parent designs with $N/2$ runs. 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 does not impact the quality of the resulting concatenated design. 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 strength-3 parent designs for our procedure must be even. Since sign switches and column permutations in the parent designs do not change their even nature, m-factor concatenated designs are also even when $m \geq N/6$. A parent design provides at most $N/4 - 1$ degrees of freedom for 2FIs. Therefore, m-factor concatenated designs provide at most $2(N/4 - 1) = N/2 - 2$ degrees of freedom for 2FIs. Concatenated even designs become even-odd only after adding the indicator factor. The m 2FIs involving this factor can be estimated independently from all other 2FIs. Hence, the maximum number of degrees of freedom for 2FIs is $N/2 - 2 + m$ when $m \geq N/6$.

Concatenated designs with added indicator factors become second-order saturated (SOS, Cheng, Mee, and Yee 2008) if the number of main effects, $m + 1$, plus the number of degrees of freedom for 2FIs 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 $m < N/4$ 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 the column change (CC) algorithm embedded within a variable neighborhood search (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.

3.2 Column Change Algorithm

Algorithm 1 shows our CC algorithm, which is a local search 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 2FI contrast matrix from scratch for every change applied by the algorithm. A detailed account of our update procedures is given in Section A of the supplementary materials. 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 11–19 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 improved design replaces the original and 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 D_l 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 D_l, and it ends with an evaluation of the concatenated design resulting from a sign switch of column m of the current plan for D_l.

Each time a sign switch of a certain column i or a swap of it with one of the (possibly sign-reversed) 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.

Algorithm 1: Pseudocode of the CC algorithm.

Input: D_u and D_l
1 $D_l^\star \leftarrow D_l$
2 Set $i \leftarrow 1$
3 **repeat**
4 **for** $i = 1, \ldots, m$ **do**
5 Construct plan D_l' by switching signs in column i of $D_l^\star$.
6 **if** *concatenated design* (D_u, D_l') *is better than* $(D_u, D_l^\star)$ **then**
7 $D_l^\star \leftarrow D_l'$
8 **else**
9 Set $j \leftarrow i + 1$
10 Set *no_improvement* $\leftarrow$ True
11 **while** $j \leq m$ and *no_improvement* **do**
12 Construct plan D_l^+ by swapping columns i and j of $D_l^\star$.
13 Construct plan D_l^- by switching the signs in column j of $D_l^\star$ and swapping the resulting column with column i.
14 Evaluate the concatenated designs (D_u, D_l^+) and (D_u, D_l^-).
15 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.
16 **if** *concatenated design* (D_u, D_l') *is better than* $(D_u, D_l^\star)$ **then**
17 $D_l^\star \leftarrow D_l'$
18 *no_improvement* $\leftarrow$ False
19 $j \leftarrow j + 1$
20 **until** *no change in* $D_l^\star$;
Output: Improved plan $D_l^\star$

3.3 Variable Neighborhood Search Algorithm

Variable neighborhood search or VNS is a metaheuristic introduced by Hansen and Mladenović (2001) as an improvement over local search algorithms for combinatorial optimization. A weakness of a local search 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 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 tuning parameters. Moreover, Hansen, Mladenović, and Moreno Pérez (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, Hertz, and Zufferey 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, Goos, and Sörensen (2009) proposed a VNS algorithm to compute D-optimal run orders for response surface designs in the presence of serial correlation. More recently, Sartono, Goos, and Schoen (2015) and Syafitri, Sartono, and Goos (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 D_l so as to minimize 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

Table 1. 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

algorithm described in Section 3.2 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, it 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, Mladenović, and Moreno Pérez 2008).

The four neighborhood structures used by our CC/VNS algorithm are listed in Table 1 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 D_l. Table 1 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 lower parent design D_l. The algorithm begins by generating a starting plan for D_l in three steps; see lines 1 and 2 of Algorithm 2. 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 are randomly permuted. Third, the resulting plan is optimized by the CC algorithm described in Section 3.2.

After the starting plan of the lower design D_l has been generated, the CC/VNS algorithm continues 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

Algorithm 2: Pseudocode of the CC/VNS algorithm.

Input: D_u and D_l
1 $R_l \leftarrow$ Random plan for D_l
2 Generate starting plan D_l^* using the CC algorithm and D_u and R_l as input.
3 Set $i \leftarrow 1$
4 **while** $i \leq 4$ **do**
5 Set *improvement* $\leftarrow$ False
6 **repeat**
7 Randomly select a plan S_l from $N_i(D_l^*)$.
8 Generate an improved plan D_l' using the CC algorithm and D_u and S_l as input.
9 **if** *concatenated design* (D_u, D_l') *is better than* (D_u, D_l^*) **then**
10 $D_l^* \leftarrow D_l'$
11 *improvement* $\leftarrow$ True
12 $i \leftarrow 0$
13 **until** *no unexplored plans left in* $N_i(D_l^*)$ or *improvement*;
14 $i \leftarrow i + 1$
Output: concatenated design (D_u, D_l^*)

structure of the improved plan. If the exploration of the second neighborhood structure does not produce any improved plan, the third neighborhood structure (N_3) is explored, and, if that does not lead to any improved plan, 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 is included in the supplementary materials of this article. Matlab allows parallel computations to decrease the calculation time. For specific parent designs, we present a comprehensive evaluation of the neighborhood structures of the CC/VNS algorithm and the computing times in supplementary Section 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.

3.4 Performance of the CC/VNS Algorithm

In this section, we first evaluate the performance of our CC/VNS algorithm to generate the best 32- and 64-run concatenated designs with up to 11 factors from strength-3 regular designs with 16 and 32 runs. To this end, we compare our concatenated designs to those obtained by Li and Lin (2016). Next, we assess

the performance of our algorithm to generate the best 128-run concatenated designs with up to 30 factors from regular and nonregular designs of strength 3 with 64 runs.

3.4.1 Comparison With a Benchmark Approach

We first evaluate 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), with the strength-3 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 2^{m-p} designs with 16 runs and up to 8 factors and regular resolution IV 2^{m-p} designs with 32 runs and up to 11 factors, respectively. They used the same design as upper parent design D_u and as lower parent design D_l and searched for a plan for D_l that sequentially minimizes the CFV of the concatenated design. For this reason, we now focus on the minimization of the F_4 vector, which is the most important component of the CFV for strength-3 designs. 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 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, for 10 factors or more, they used a large number of randomly chosen permutations instead of all possible permutations. They ran their enumeration program for 10- and 11-factor parent designs for 168 hr 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.

The results of 1000 iterations of the CC/VNS algorithm applied to the cases of LL16 are shown in Table 2. The computing times for the 6-, 7-, and 8-factor designs were less than a second per iteration on a standard CPU (Intel(R) Core(TM) i7 processor, 2.8 GHz, 8 GB). For 9 factors or more, the computing times ranged from 1 to 6 sec per iteration.

Table 2. Results produced by the CC/VNS algorithm for 32- and 64-run concatenated designs constructed from regular resolution IV designs. The labels of the parent designs are those used by Chen, Sun, and Wu (1993). $F_4^{\max}$ represents the frequency of the largest values of the J_4-characteristics. 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 1000 iterations.

N	Parent	$F_4^{\max}$	Percentage of iterations	Percentage of plans
32	6-2.1	4	100	134.104
	7-3.1	12	100	18.953
	8-4.1	24	100	2.132
64	7-2.1	0	88.1	24.580
	7-2.2	0	100	27.421
	7-2.3	4	100	39.416
	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
	9-4.1	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
	10-5.1	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
	11-6.1	42	100	0.002
	11-6.2	44	65.9	0.003

The first column of the table shows the run size N of the concatenated design. The second column shows the parent designs used for the concatenation in the form m-$p.z$, where m is the number of factors, p is the number of generators of the design, and z is the ranking of the design according to the aberration criterion (Chen, Sun, and Wu 1993). The third column presents the frequencies $F_4^{\max}$ of the largest values of the J_4-characteristics of the concatenated designs. The largest J_4-characteristic equals 16 for the best 32-run designs we found and 32 for the best 64-run designs, except for the designs constructed from the parent designs 7-2.1 and 7-2.2. These parent designs resulted in 64-run designs in which all J_4-characteristics equal zero. As a result, all designs we found have a generalized resolution of at least 4.5. None of the 32-run designs we obtained have J_4-characteristics of 32, so that there is no complete aliasing between 2FIs. 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. So, there is also no complete aliasing between 2FIs in the 64-run designs. The F_4 vectors 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, $F_4(32) = 44$. So, in our design, there is less aliasing between the 2FIs.

The fourth column of Table 2 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 2, 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, 10-5.4, and 11-6.2, the percentages were 88.1, 96.8, and 65.9, respectively. These large percentages imply that it is almost certain that 10 iterations of the CC/VNS algorithm suffice to find the best concatenated design. As a matter of fact, even for the worst case in Table 2, the probability to obtain the best design at least once in 10 iterations is $1 - (1 - 0.659)^{10} \approx 1$. Remarkably, in each iteration where the CC/VNS algorithm failed to find the best design with $F_4(32) = 44$ for this case, it produced LL16's design with $F_4(32) = 46$. This shows that, even when the algorithm fails to find the best design, it produces a high-quality alternative.

Column 5 of Table 2 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 by the CC/VNS algorithm. The exception is parent design 6-2.1, for which the CC/VNS algorithm evaluated 34% more plans than a complete enumeration would ($6! \times 2^2 = 2880$). However, in each iteration of the CC/VNS algorithm, the CC algorithm provided a starting plan with an optimal F_4 vector. Constructing this plan only required 2.35% of the total number of evaluations. The additional computations are due to the CC/VNS algorithm's visits to the four neighborhood structures, to confirm the excellent quality of the starting solution produced by the CC algorithm.

Our comparison with the benchmark approach and the comprehensive evaluation of the CC/VNS algorithm in Section B

of the supplementary materials allows us to conclude that our CC/VNS algorithm creates high-quality concatenated designs using a considerably smaller computing effort than that needed by LL16.

3.4.2 Performance for 128-Run Designs

We also tested the potential of the CC/VNS algorithm by constructing 128-run designs involving 10, 15, 20, 25, and 30 factors from 64-run parents. We obtained suitable 10-, 15-, 20-, 25-, and 30-factor parent designs from the complete collection of regular 64-run resolution IV 2^{m-p} designs (Chen, Sun, and Wu 1993), the collection of nonregular designs based on quaternary linear codes (QLC) found 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 (Sloane 1999).

Supplementary Section B shows our detailed results for 100 iterations of the CC/VNS procedure, for the case where we optimized the B_4 value of the concatenated designs as well as for the case where we optimized the F_4 vector. When optimizing the B_4 value of designs with up to 20 factors, the best design was found in 65% or more of the iterations of the CC/VNS algorithm. The cases with 25 and 30 factors were clearly more challenging, as the success rate dropped to values as low as 6% and 13% for these cases. When optimizing the F_4 vector, the success rates are even lower than that, because optimizing the F_4 vector is harder than optimizing the B_4 value. The small probability of identifying the best concatenated design is not a major problem, because, whenever it fails to find the best design, the CC/VNS algorithm still produces a high-quality concatenated design with the same generalized resolution as the best one and with a B_4 value and an F_4 vector that are only slightly worse than those of the best concatenated design. From our results, we concluded that 10 iterations will generally suffice to find a high-quality 128-run design. In the event there are no more than 20 factors, that design will most likely be the best.

For all cases studied here, only a very small proportion (generally much smaller than 1%) of all possible plans for the lower parent design D_l are evaluated by the CC/VNS algorithm when constructing the concatenated design. The computing times for optimizing the B_4 value ranged from 1 to 709 sec for one iteration of the algorithm. For optimizing the F_4 vector, the computing times varied from 2 to 904 sec per iteration, for up to 20 factors. For 25 and 30 factors, the computing times went up to 3.3 hr.

3.5 Choice of Parent Designs

To investigate how the quality of concatenated designs depends on the choice of the parent designs, we constructed 64- and 128-run concatenated designs with parent designs that differ in G- or G_2-aberration. The parent designs we used here to construct 64-run designs were selected from the complete catalog of 32-run strength-3 designs (Schoen, Eendebak, and Nguyen 2010) with 8, 10, 12, 14, and 16 factors, while the parent designs we used to construct 128-run designs were selected from the complete catalog of regular 64-run resolution IV 2^{m-p} designs (Chen, Sun, and Wu 1993) with 16, 18, and 20 factors. In this section, we discuss the results for concatenated designs with 10 factors and 64 runs, as well as those for concatenated designs with 16 factors and 128 runs. Results for all other cases follow the same pattern and allow for the same conclusions.

The complete catalog of 32-run designs with 10 factors includes 32 designs. Their B_4 values range from 10 to 18 and the frequencies of the J_4-characteristics of 32 range from 1 to 18. The complete catalog of 64-run regular designs with 16 factors includes 48 designs. Their B_4 values and, equivalently, the frequencies of the J_4-characteristics of 64 range from 43 to 105. From each of these two catalogs, we selected a set of five parent designs to construct F_4-optimized concatenated designs and a set of five parent designs for B_4-optimized concatenated designs. For B_4-optimized concatenated designs, the selected parent designs were the best and worst designs according to the G_2-aberration criterion and those corresponding to the first, second, and third quartiles in that ranking. For minimizing the F_4 value, we selected the designs in a similar fashion based on the G-aberration criterion. We concatenated all 15 possible pairs of the five selected parent designs, including pairs of the same designs.

Table 3 shows the results for 100 iterations of the CC/VNS algorithm when the objective is to minimize the B_4 value. The table shows that the smallest B_4 values in the 64- and 128-run concatenated designs result from concatenating two copies of the minimum G_2-aberration designs. Concatenating minimum G_2-aberration designs with any of the four other designs results in larger B_4 values for the concatenated designs. Table 3 also shows that using bad 32- and 64-run designs in terms of the G_2-aberration criterion, such as the worst design (W) and the design corresponding with the third quartile (Q_3) of the G_2-aberration ranking, generally results in 64- and 128-run designs with the largest B_4 values. It is interesting to mention though that the 64-run concatenated design constructed from the 32-run parent designs B and Q_2 has a better B_4 value than the concatenated design constructed from the parent designs B and Q_1, even though design Q_2 has a worse G_2-aberration than design Q_1. This implies that the G_2-aberration ranking of the

Table 3. B_4 values for the B_4-optimized concatenated designs with 64 and 128 runs. The symbols B, Q_1, Q_2, Q_3, and W correspond to the best ranked design, the designs corresponding to the first, second, and third quartiles, and the worst ranked design, respectively, in terms of G_2-aberration.

(a) 10 factors and 64 runs

Parents	B	Q_1	Q_2	Q_3	W
B	4				
Q_1	5.69	6			
Q_2	5.5	6.19	6		
Q_3	5.63	6.31	6.38	6.38	
W	6	6.69	7	7.13	7.5

(b) 16 factors and 128 runs

Parents	B	Q_1	Q_2	Q_3	W
B	17				
Q_1	21.75	20			
Q_2	22	25.25	25		
Q_3	23	26.25	26	27	
W	32.5	33.75	36.5	37.5	45

Table 4. Frequencies of J_4-characteristics of 32 and 64 for the F_4-optimized concatenated designs with 64 and 128 runs, respectively. The symbols B, Q_1, Q_2, Q_3, and W correspond to the best ranked design, the designs corresponding to the first, second, and third quartiles, and the worst ranked design, respectively, in terms of G-aberration.

(a) 10 factors and 64 runs

Parents	B	Q_1	Q_2	Q_3	W
B	0				
Q_1	2	4			
Q_2	4	6	8		
Q_3	7	9	12	16	
W	14	15	19	24	30

(b) 16 factors and 128 runs

Parents	B	Q_1	Q_2	Q_3	W
B	72				
Q_1	89	96			
Q_2	90	105	108		
Q_3	94	109	110	116	
W	134	147	150	156	196

parent designs does not necessarily agree with the ranking of the resulting concatenated designs in terms of the B_4 value.

Table 4 shows the results for 100 iterations of the CC/VNS algorithm when the objective is to minimize the F_4 vector. The conclusions that can be drawn from that table are similar to those for the B_4-optimized concatenation. That is, better parent designs in terms of the F_4 vector lead to better F_4 vectors for the concatenated designs. For instance, the table shows that concatenating two copies of the best parent design with 32 runs leads to a 64-run concatenated design without any J_4-characteristic of 32 and a generalized resolution as large as 4.75. For the 128-run design case, concatenating two copies of the best 64-run regular design leads to a much lower frequency of the J_4-characteristics of 64 than concatenating any other pair of selected parents.

The specific cases discussed here as well as our more comprehensive study of the relation between the quality of concatenated designs and the choice of the parent designs permit the following conclusion: Concatenating the best parent designs in terms of the G- and G_2-aberration generally leads to the best concatenated designs in terms of the F_4 vector and B_4 value, respectively, within the class of designs that can be obtained by concatenating strength-3 designs. For this reason, when constructing concatenated designs using our CC/VNS algorithm, we recommend to use the best parent designs available in terms of the desired optimization criterion.

4. Results

Encouraged by the test results, we used the CC/VNS algorithm to generate two-level even-odd designs for run sizes 64 and 128, based on the best parent designs available with 32 and 64 runs, respectively. A detailed description of the parent designs is included in supplementary Section C. Tables showing the sign switches and permutations in the lower parent design along with the full F_4 vectors of the concatenated designs are presented in supplementary Section D. In the present section, we discuss the most important features of the concatenated designs and compare them with benchmark designs from the literature.

For each combination of number of runs and number of factors, we considered several pairs of attractive parent designs. 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, especially when the number of factors exceeds 20 (see supplementary Section A). After running the CC/VNS algorithm, we constructed our final k-factor design by adding the indicator factor to the resulting m-factor concatenated design. Section D of the supplementary materials shows that nine of our best 66 concatenated designs are constructed from different, nonisomorphic parent designs. In all other cases, the upper and lower parent designs were isomorphic.

4.1 64 Runs

Table 5 shows the B_4 value, the generalized resolution (GR), the frequency of the largest J_4-characteristic ($F_4^{\max}$), and the degrees of freedom for two-factor interactions (df) of 48 64-run designs involving 9–17 factors. It should be pointed out that comparing the $F_4^{\max}$ values of two designs only make sense if they have the same GR value and thus the same maximum J_4-characteristic. In that case, the design with the smaller $F_4^{\max}$ value has the better F_4 vector of the two.

The tabulated results are for the best concatenated designs the CC/VNS algorithm produced in terms of the F_4 vector (denoted by CC/F4) and in terms of the B_4 value (denoted by CC/B4). The parent designs we used as inputs were the best designs from the enumeration of Schoen, Eendebak, and Nguyen (2010); see supplementary Section C. The table also includes the following benchmark designs:

- The regular designs listed by Chen, Sun, and Wu (1993). The designs included in the table have the smallest possible B_4 value and, subject to this, the largest number of degrees of freedom for 2FIs. Therefore, they are either minimum aberration designs (denoted by MA) or alternative designs with a larger number of degrees of freedom for 2FIs than the MA design (denoted by EST).
- The QLC designs of Xu and Wong (2007). We denote the QLC design with the best B_4 value as QLC/B4 and the design with the best F_4 vector as QLC/F4. We use the label QLC whenever there is a single design that is best in terms of the B_4 value and in terms of the F_4 vector.
- The 17-factor design of Cheng, Mee, and Yee (2008), denoted by CMY.
- The best projections of the folded-over 32-run Paley matrix.

The third column of Table 5 shows whether the designs are admissible (Sun, Wu, and Chen 1997), meaning that they are not dominated by another design when considering the B_4 value, the generalized resolution, the F_4 vector, and the degrees of freedom for 2FIs. A design that is dominated by another design is called inadmissible. For example, the 9-factor MA design has the same B_4 value as the CC/B4 design, but its GR value and number of degrees of freedom for 2FIs are lower (note that the $F_4^{\max}$ value for the MA design should not

Table 5. 64-run designs involving 9–17 factors. B_4 values rounded to nearest integer. CC/B4: design by CC/VNS under B_4 optimization; CC/F4: design by CC/VNS under F_4 optimization; MA: regular minimum aberration design; EST: regular design with same B_4 value as MA design, but larger number of degrees of freedom for 2FIs; QLC: design based on quaternary linear code with best B_4 value and F_4 vector; QLC/B4: QLC design with best B_4 value; QLC/F4: QLC design with best F_4 vector; CMY: design from Cheng, Mee, and Yee (2008); P: projection of folded-over 32-run Paley matrix; [a]: design permits estimation of all 2FIs; [s]: SOS design.

k	Construction	Admissible	B_4	GR	$F_4^{\max}$	df	k	Construction	Admissible	B_4	GR	$F_4^{\max}$	df
9	CC/B4[a]	Yes	1	4.75	16	36	14	CC/B4	No	22	4.5	88	43
	MA	No	1	4	1	33		EST	No	22	4	22	45
	QLC[a]	No	1	4.5	4	36		QLC[s]	Yes	14	4.5	56	49
	P	No	4	4.75	58	31		P	Yes	33	4.75	526	31
	CC/F4[a]	Yes	1	4.75	16	36		CC/F4	Yes	24	4.5	24	43
10	CC/B4[a]	Yes	2	4.75	32	45	15	CC/B4	Yes	33	4.25	8	44
	MA	No	2	4	2	39		MA	Yes	30	4	30	43
	QLC	No	2	4.5	8	39		QLC	No	33	4	21	43
	P	No	6	4.75	96	31		P	Yes	45	4.75	726	31
	CC/F4	No	2	4.75	32	44		CC/F4	Yes	35	4.5	38	44
11	CC/B4	Yes	4	4.5	16	48	16	CC/B4	Yes	45	4	9	45
	MA	No	4	4	4	44		MA	Yes	43	4	43	43
	QLC	No	4	4.5	16	47		QLC/B4	No	47	4	31	43
	P	No	10	4.75	160	30		QLC/F4	No	60	4	28	31
	CC/F4	Yes	7	4.75	108	40		P	Yes	61	4.75	978	31
12	CC/B4	Yes	10	4.5	21	41		CC/F4	Yes	49	4.5	57	45
	MA	Yes	6	4	6	50	17	CC/B4[s]	Yes	60	4	12	46
	QLC	Yes	6	4.5	24	48		CMY[s]	No	60	4	28	46
	P	Yes	16	4.75	252	31		MA	Yes	59	4	59	43
	CC/F4	Yes	11	4.5	5	41		QLC/B4	Yes	59	4	59	43
13	CC/B4	Yes	15	4.5	36	42		QLC/F4	No	64	4	40	43
	EST[s]	Yes	14	4	14	50		P	Yes	80	4.75	1286	31
	QLC	Yes	10	4.5	40	48		CC/F4[s]	Yes	65	4.5	83	46
	P	Yes	23	4.75	370	31							
	CC/F4	Yes	16	4.5	10	42							

be compared to that of the CC/B4 design, because the GR value and thus also the maximum J_4-characteristic are different for the two designs). Therefore, the 9-factor MA design is inadmissible.

Overall, 31 of the 48 64-run designs under comparison are admissible. Table 5 shows that all but two of the designs produced by the CC/VNS algorithm are admissible. The 10-factor CC/F4 design is inadmissible because it is dominated by the CC/B4 design, while the 14-factor CC/B4 design is inadmissible because it is dominated by the QLC design. There are seven inadmissible QLC designs, all of which are dominated by the CC/B4 designs. Three regular MA designs are inadmissible as they are dominated by the CC/B4 design, and one regular EST design is inadmissible because it is dominated by the QLC design. The 9-, 10-, and 11-factor designs based on the folded-over Paley matrix are dominated by the CC/F4 designs. Finally, the 17-factor CMY design is dominated by the CC/B4 design.

Using the folded-over 32-run Paley matrix always results in the largest generalized resolution of 4.75, but, for 9, 10, and 11 factors, the CC/B4 and/or the CC/F4 design have the same generalized resolution, a better F_4 vector, and a larger number of degrees of freedom for 2FIs. For larger numbers of factors, the CC/VNS algorithm does not produce designs with that large a generalized resolution, because the CC/VNS algorithm involves strength-3 parent designs and the folded-over Paley matrix is essentially a concatenation of two strength-2 designs.

For 9 and 10 factors, the B_4 values we obtained are the minimum values possible (Xu 2005). For 12–14 and 17 factors, the B_4 values of the designs produced by the CC/VNS algorithm are larger than those of the QLC designs and/or those of the regular MA or EST designs. This might imply that we did not use the best possible input designs for the CC/VNS algorithm. However, it is impossible to split the 64-run QLC designs with 12–14 and 17 factors into two strength-3 designs, so that these designs cannot be constructed by concatenating two strength-3 32-run designs. Likewise, it is not possible to split the regular resolution IV designs with 12, 13, 15, 16, and 17 factors into two strength-3 32-run designs. Therefore, the fact that the CC/VNS algorithm did not produce designs with lower B_4 values should not be viewed as a weakness of the algorithm, but as a consequence of our focus on concatenating strength-3 designs.

The column labeled df in Table 5 shows that the CC/VNS algorithm produces designs with 9–11 and 15–17 factors that provide at least as many degrees of freedom for 2FIs as the best benchmark designs. For 12–14 factors, the numbers of degrees of freedom for 2FIs of the CC/B4 and CC/F4 designs are smaller than those of the regular designs and the QLC designs. However, the numbers of degrees of freedom for 2FIs of the concatenated designs with 12–14 factors are the maximum numbers obtainable by concatenating even parent designs; see Section 3.1. As there are no even-odd strength-3 parent designs with 32 runs and more than 10 factors, it is not possible to construct concatenated designs with larger numbers of degrees of freedom for 2FIs for these cases when using strength-3 designs as parent designs.

Table 5 also identifies designs with which all 2FIs are estimable as well as SOS designs. The CC/VNS algorithm produced 9- and 10-factor designs which allow all 2FIs to be estimated. Our 17-factor designs, the 17-factor design of Cheng, Mee, and Yee (2008), the 13-factor regular resolution IV design, and the 14-factor QLC design of Xu and Wong (2007) are SOS designs. The 13- and 14-factor SOS designs cannot be constructed by concatenating two strength-3 designs.

4.2 128-Run Designs

The parent designs we used for the CC/B4 designs with 128 runs were the 64-run regular minimum aberration designs (Chen, Sun, and Wu 1993), the 64-run QLC designs (Xu and Wong

2007), and our own 64-run designs that minimize the B_4 value. For the 128-run concatenated designs that optimize the F_4 vector, the parent designs we used were the best projections of the folded-over 32-run Paley matrix and the 64-run concatenated designs produced by the CC/VNS algorithm. A detailed report of the best 128-run designs we obtained and their parent designs is given in Sections C and D of the supplementary materials.

4.2.1 10–15 Factors

It is known that strength-4 128-run designs exist with up to 15 factors; see Hedayat, Sloane, and Stufken (1999) for the construction of the 15-factor design. These designs necessarily consist of two concatenated strength-3 64-run designs augmented with an indicator factor. Therefore, provided the right strength-3 64-run parent designs are used as input, the CC/VNS algorithm should be able to construct strength-4 128-run designs. This proved to be the case for our CC/B4 designs with 10–15 factors and our CC/F4 designs with 10 and 11 factors constructed using QLC and CC/B4 parent designs with 64 runs. For 12–15 factors, our CC/F4 designs only have a resolution of 4.75. The parents of these strength-3 designs are the best projections of the folded-over 32-run Paley matrix and our CC/F4 designs with 64 runs.

Supplementary Section E provides a comprehensive discussion of the strength-4 designs. It compares the 10- and 11-factor regular resolution V designs, the QLC designs involving 10–15 factors and the minimum G-aberration designs we identified based on the complete catalog of 128-run strength-4 OAs produced by Schoen, Eendebak, and Nguyen (2010). To the best of our knowledge, we are the first to identify the minimum G-aberration 128-run designs of strength 4.

4.2.2 16–33 Factors

Table 6, which has the same format as Table 5, shows the main results for designs with 16–33 factors. The table includes our own concatenated designs as well as the following benchmark designs:

- The regular designs listed by Xu (2009). The designs we used as benchmarks have the smallest possible B_4 value and, subject to this, the largest number of degrees of freedom for 2FIs. Therefore, they are either MA or EST designs.
- The QLC designs of Xu and Wong (2007).

Table 6 includes 84 designs, 19 of which are inadmissible. Seven of the 36 designs produced by the CC/VNS algorithm are inadmissible. The 16-factor CC/B4 design is dominated by the QLC/F4 design. The CC/B4 designs involving 17, 22–24, 26, and 27 factors are dominated by the corresponding QLC/B4 designs. The regular (MA or EST) designs for 16, 17, 20, 21, and 22 factors are dominated by the corresponding QLC/B4 designs and the 26-, 27-, and 28-factor regular designs are dominated by the corresponding QLC designs. Four QLC designs are inadmissible: the QLC/B4 designs for 29 and 30 factors are dominated by the MA designs, while the QLC/F4 designs for 21 and 30 factors are dominated by the CC/B4 designs. The table further suggests that the QLC/F4 design with 30 factors is also dominated by the CC/F4 design, but this is due to the rounding of the B_4 value to the nearest integer.

All CC/F4 designs outperform all benchmark designs as well as the CC/B4 designs in terms of generalized resolution. Therefore, all CC/F4 designs are admissible. Except for the 12-factor case, the best parent designs for the CC/F4 designs are the 64-run designs generated from projections of the folded-over

Table 6. 128-run designs involving 16–33 factors. B_4 values rounded to nearest integer. CC/B4: design by CC/VNS under B_4 optimization; CC/F4: design by CC/VNS under F_4 optimization; MA: regular minimum aberration design; EST: regular design with same B_4 value as MA design, but larger number of degrees of freedom for 2FIs; QLC: design based on quaternary linear code with best B_4 value and F_4 vector; QLC/B4: QLC design with best B_4 value; QLC/F4: QLC design with best F_4 vector; ^{s}: SOS design.

k	Construction	Admissible	B_4	GR	$F_4^{\max}$	df	k	Construction	Admissible	B_4	GR	$F_4^{\max}$	df	k	Construction	Admissible	B_4	GR	$F_4^{\max}$	df
16	CC/B4	No	12	4	2	94	22	CC/B4	No	90	4	46	83	29	CC/B4	Yes	330	4	38	90
	MA	No	10	4	10	90		MA	No	65	4	65	102		MA^s	Yes	266	4	266	98
	QLC/B4	Yes	10	4	2	90		QLC/B4	Yes	56	4	42	105		QLC/B4	No	290	4	290	87
	QLC/F4	Yes	11	4.5	32	98		QLC/F4	Yes	66	4	28	96		QLC/F4	Yes	315	4	123	89
	CC/F4	Yes	19	4.75	131	77		CC/F4	Yes	97	4.75	753	83		CC/F4	Yes	343	4.75	2800	90
17	CC/B4	No	17	4	6	99	23	CC/B4	No	110	4	110	83	30	CC/B4	Yes	386	4	70	91
	EST	No	15	4	15	95		MA	Yes	83	4	83	102		MA	Yes	335	4	335	87
	QLC/B4	Yes	15	4	3	102		QLC	Yes	83	4	36	97		QLC/B4	No	336	4	336	87
	QLC/F4	Yes	16	4.5	48	99		CC/F4	Yes	119	4.75	942	84		QLC/F4	No	396	4	145	89
	CC/F4	Yes	28	4.75	189	78	24	CC/B4	No	136	4	76	85		CC/F4	Yes	396	4.75	3280	91
18	CC/B4	Yes	23	4	11	99		EST	Yes	102	4	102	102	31	CC/B4	Yes	447	4	43	92
	EST	Yes	20	4	20	97		QLC	Yes	101	4	45	97		MA	Yes	391	4	391	87
	QLC/B4	Yes	20	4	4	93		CC/F4	Yes	145	4.75	1150	85		QLC/B4	Yes	391	4	391	87
	QLC/F4	Yes	24	4.5	64	92	25	CC/B4	Yes	165	4	51	86		QLC/F4	Yes	417	4	161	91
	CC/F4	Yes	37	4.75	263	79		MA^s	Yes	124	4	124	102		CC/F4	Yes	458	4.75	3796	92
19	CC/B4	Yes	30	4	14	100		QLC	Yes	123	4	55	98	32	CC/B4	Yes	517	4	60	93
	MA	Yes	27	4	27	99		CC/F4	Yes	175	4.75	1405	86		MA	Yes	452	4	452	87
	QLC/B4	Yes	25	4	15	103	26	CC/B4	No	198	4	109	87		QLC/B4	Yes	452	4	452	87
	QLC/F4	Yes	32	4.5	96	98		MA	No	152	4	152	98		QLC/F4	Yes	480	4	192	91
	CC/F4	Yes	48	4.75	355	80		QLC	Yes	146	4	66	98		CC/F4	Yes	528	4.75	4372	93
20	CC/B4	Yes	40	4	20	105		CC/F4	Yes	209	4.75	1695	87	33	CC/B4^s	Yes	592	4	52	94
	MA	No	36	4	36	99	27	CC/B4	No	237	4	237	88		MA	Yes	518	4	518	87
	QLC/B4	Yes	32	4	18	103		MA	No	180	4	180	98		QLC/B4	Yes	518	4	518	87
	QLC/F4	Yes	39	4	15	95		QLC	Yes	174	4	78	99		QLC/F4	Yes	540	4	220	93
	CC/F4	Yes	61	4.75	457	81		CC/F4	Yes	248	4.75	2018	88		CC/F4^s	Yes	606	4.75	5044	94
21	CC/B4^s	Yes	52	4	20	106	28	CC/B4	Yes	280	4	69	89							
	MA	No	51	4	51	102		MA	No	210	4	210	98							
	QLC/B4	Yes	42	4	28	105		QLC^s	Yes	203	4	91	99							
	QLC/F4	No	52	4	21	96		CC/F4	Yes	293	4.75	2386	89							
	CC/F4	Yes	78	4.75	584	82														

32-run Paley matrix. So, all but one of the CC/F4 designs are constructed from a Paley-based design. If we denote the number of factors from that parent design by m, then we can verify in the column labeled df in Table 6 that the degrees of freedom for 2FIs is $2 \times 31 + m$ for each CC/F4 design.

The fact that our CC/B4 designs involving 16, 17, 22–24, 26, and 27 factors are inadmissible is not due to a bad choice of strength-3 parent designs. To reach this conclusion, we verified that none of the 128-run QLC designs can be split in two 64-run strength-3 designs according to one of their factors. Therefore, the 128-run k-factor QLC designs, say, cannot be constructed by concatenating two strength-3 designs with $k - 1$ factors and adding the indicator factor. If it were possible to construct k-factor 128-run QLC designs by concatenating two 64-run strength-3 parents, then these parents should also involve k factors. Now, 64-run strength-3 parent designs involving 22 factors or more are necessarily even designs. Since concatenating two even designs results in a design that is even too, the QLC designs for 22 factors or more, which are even-odd, cannot be constructed by concatenating two strength-3 designs. For the 16- and 17-factor cases, even-odd parent designs do exist. If a construction by concatenation of the strength-3 QLC designs were possible, we should be able to extend the 16- and 17-factor designs with an extra factor that indicates the parent designs, and the extended designs should also have a strength of 3. We tried to extend the 16-factor and 17-factor designs using the algorithm of Schoen, Eendebak, and Nguyen (2010), but it did not produce an extension in 6 hr of computing time, while it normally takes less than a second to extend similar designs with 64 runs. For this reason, we conjecture that the 128-run QLC designs with 16 and 17 factors cannot be constructed by concatenating strength-3 designs. So, the reason why many of our CC/B4 designs are inadmissible is that we restrict ourselves to strength-3 parent designs and not that the CC/VNS algorithm performs poorly.

Table 6 shows that we were able to find designs with 18, 20, 21, and 30–33 factors which provide more degrees of freedom for 2FIs than the benchmark designs. The two 33-factor designs we obtained and one of our 21-factor designs 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.

The CC/VNS algorithm enabled us to add 11 admissible CC/B4 designs and 18 admissible CC/F4 designs to the literature on 128-run designs. We also found that certain designs from the literature are inadmissible when considering the generalized resolution, the B_4 value, the F_4 vector, and the degrees of freedom for 2FIs.

5. Practical Examples

We now return to the enzyme stability experiment and the software process simulation experiment that motivated this article. For each of the two experiments, we explore several 64- and 128-run design options.

5.1 The Enzyme Stability Experiment

The goal of the enzyme stability experiment was to improve the stability of an enzyme in a watery solution at room temperature. There were 17 experimental factors, which indicated the presence or absence of 17 possible additives. The experiment involved small test tubes with the enzyme and the additives. The tubes were stored at room temperature for 8 weeks, and sampled at the start of the study and after 15, 30, and 60 days to check enzyme activity. A total of 64 combinations of the stabilizer was practically feasible, so that a 64-run design was used.

Table 7 shows 11 design options for the enzyme stability experiment. Designs with 64 runs available in the literature were the regular 64-run MA designs (Chen, Sun, and Wu 1993), the QLC designs (Xu and Wong 2007), and the 17-factor design of Cheng, Mee, and Yee (2008), denoted as CMY in the table. We were reluctant to use the designs from the literature because they have 28 or more J_4-characteristics of 64, so that many pairs of 2FIs are completely aliased. Via an ad hoc procedure, we derived design X, which has only three J_4-characteristics of 64. Its complete F_4 vector is $F_4(64, 48, 32, 16) = (3, 6, 125, 625)$. So all the 64-run designs from the literature as well as design X have a generalized resolution of 4. Design X was the one eventually used for the experiment.

Table 7 includes three 64-run candidate designs for the enzyme stability experiment from our work for the present article: the CC/B4 design, the CC/F4 design, and a design obtained by folding over the 32-run Paley matrix (design P in the table). It turns out that the 64-run CC/F4 design dominates design X, so that the ad hoc design actually used is inadmissible. The CMY design is also inadmissible because it is dominated by our CC/B4 design. In hindsight, we should perhaps have opted for the design obtained by folding over the 32-run Paley matrix, because of its large generalized resolution and the implication that the maximum J_4-characteristic value is only 16 for that design. The Paley-based design provides 31 degrees of freedom for estimating 2FIs. This number compares rather poorly with the designs from the literature and with the CC/VNS designs. However, for the enzyme stability case, substantially fewer important interactions than 31 were expected, so that 31 degrees of freedom for estimating 2FIs would have been amply sufficient.

When discussing supersaturated designs, Marley and Woods (2010) argued that the degrees of freedom available for model selection should be at least half the number of eligible terms. As the strength-3 designs studied here are supersaturated for the 2FIs, heeding the advice of these authors in an unrestricted search for 2FIs (i.e., without imposing heredity) would require at least $\binom{17}{2}/2 = 68$ degrees of freedom for the 2FIs. Obviously,

Table 7. Design options for the 17-factor enzyme stability experiment.

N	Design	B_4	GR	$F_4^{\max}$	df
64	X	76	4	3	34
	CMY	60	4	28	46
	MA	59	4	59	43
	QLC/B4	59	4	59	43
	QLC/F4	64	4	40	43
	CC/B4	60	4	12	46
	CC/F4	65	4.5	83	46
	P	80	4.75	1286	31
128	QLC/B4	15	4	3	102
	QLC/F4	16	4.5	48	99
	CC/F4	28	4.75	189	78

64-run experiments are not large enough to provide this number of degrees of freedom. For this reason, we also study several admissible 128-run options that are compatible with the advice of Marley and Woods (2010). These design options are shown in the last three lines of Table 7. As minimizing the correlations among the 2FI contrast vectors reduces the ambiguity in the interpretation of the results, we prefer the 128-run CC/F4 design, despite the fact that it has the smallest number of degrees of freedom for estimating 2FIs of the three 128-run design options. By construction, the 128-run CC/F4 design has an indicator factor whose interactions with the 16 other factors can be estimated independently. As the total number of degrees of freedom for estimating interactions is 78 for that design, this leaves 62 degrees of freedom for the remaining 120 2FIs. This is compatible with the advice of Marley and Woods (2010).

5.2 The Software Process Simulation Experiment

The second motivating experiment is discussed in Houston et al. (2001) and is concerned with a sensitivity analysis of a simulation model for a software process. One of the designs used had 30 factors and 64 runs. Houston et al. (2001) mentioned that it was a regular design constructed using statistical software, but they do not provide any further details. Given that complete catalogs of regular 64-run designs of strength 3 have been available since 1993 (Chen, Sun, and Wu 1993), they might have used the 64-run 30-factor minimum aberration design. Table 8 shows the properties of that option, along with those of an alternative based on the folded-over 32-run Paley matrix and several admissible 128-run options.

Both 64-run design options listed for the software process simulation experiment are constructed by folding-over a strength-2 design. Both designs provide 31 degrees of freedom for estimating 2FIs and have a B_4 value of 945. However, the MA design is inadmissible because its GR value is smaller than that of the Paley-based design. So, the latter design is a better alternative for the software process simulation experiment. Given that there are 435 2FIs when 30 factors are studied, it is impossible to analyze the interactions without imposing restrictions such as strong heredity, which implies that a 2FI should be considered for inclusion in the fitted model only if both of its parent MEs are active. One option is to conduct the analysis of the MEs and the 2FIs in two successive steps, and to impose strong heredity restrictions in the second step (Miller and Sitter 2001). The analysis would be compatible with the advice of Marley and Woods (2010) if the number of active MEs turns out to be at most 11.

Table 8 also lists three admissible design options with 128 runs. The number of degrees of freedom for estimating 2FIs of these designs is more than twice as large as that for the listed 64-run options. Therefore, under strong heredity, model selection based on the 128-run designs would be compatible with the advice of Marley and Woods (2010) if the number of active MEs identified in the first step turned to be at most 19 (8 more than for the 64-run design options). Among the 128-run designs in Table 8, we would prefer CC/F4 design because it minimizes the correlations between pairs of 2FIs contrast vectors and because it has the largest number of degrees of freedom for estimating 2FIs.

Table 8. Design options for the 30-factor software process simulation experiment.

N	Design	B_4	GR	$F_4^{\max}$	df
64	MA	945	4	945	31
	P	945	4.75	15,120	31
128	CC/B4	386	4	70	91
	MA	335	4	335	87
	CC/F4	396	4.75	3280	91

6. Discussion

In this article, 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 the two-factor interactions in the concatenated design. Using the CC/VNS algorithm, we generated two-level even-odd designs with 64 and 128 runs and up to 33 factors. Sixteen out of the 18 newly generated 64-run designs and 29 out of the 36 newly generated 128-run designs were admissible in terms of the aliasing of two-factor interactions and in terms of the degrees of freedom for estimating two-factor interactions when compared with benchmark designs from the literature.

All but one of our 64-run designs have a smaller G-aberration than the designs of Chen, Sun, and Wu (1993), Block and Mee (2005), Xu and Wong (2007), and Xu (2009). We obtained the best 64-run designs in terms of the generalized resolution from projections from the folded-over 32-run Hadamard matrix given by the Paley construction. However, a drawback of these designs is that they are even. Therefore, they provide at most 31 degrees of freedom for estimating two-factor interactions. 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.

The indicator factor with $N/2$ positive ones and $N/2$ negative ones causes even concatenated designs, produced by concatenating two even parent designs, to become even-odd, and, for $k = N/4 + 1$ factors, to be second-order saturated. When based on even-odd parent designs, concatenated designs can be even-odd without the inclusion of the indicator factor. Even-odd and SOS designs are attractive for estimating models including all the main effects and a considerable number of two-factor interactions. Alternative nonorthogonal designs to estimate interactions can be found in Li and Nachtsheim (2000), Smucker, del Castillo, and Rosenberg (2011), Smucker, del Castillo, and Rosenberg (2012), and Smucker and Drew (2015).

Selection strategies for models with main effects and interactions can be found in Draguljić et al. (2014), for instance. Alternatively, one 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 indicator factor in our concatenated designs.

The indicator factor 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 or machines, offering more flexibility for the experimentation.

If the concatenated design is made up from nonisomorphic parent designs, we recommend to run the parent design with the smallest B_4 value first, or the one with the best F_4 vector, or the largest number of degrees of freedom for estimating interactions, depending on the interest of the experimenter. There are four cases in which our 64-run designs are constructed from nonisomorphic parent designs and five cases in which our 128-run designs are constructed from nonisomorphic parents. Details are given in supplementary Section D.

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, multilevel arrays, or mixed-level arrays.

Supplementary Materials

Supplementary sections.pdf Objective functions and fast update methods; algorithm performance evaluation; parent designs; concatenated designs; 128-run designs of strength 4.
Programs.zip Matlab implementation of the CC/VNS algorithm.

Acknowledgments

The authors thank the editor, the associate editor and the referees for their valuable comments.

Funding

The research that led to this article 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. [222]

Block, R. M., and Mee, R. W. (2005), "Resolution IV Designs With 128 Runs," *Journal of Quality Technology*, 37, 282–293. [220,230]

Butler, N. A. (2004), "Minimum G_2-Aberration Properties of Two-Level Foldover Designs," *Statistics & Probability Letters*, 67, 121–132. [219]

Butler, N. A. (2007), "Results for Two-Level Fractional Factorial Designs of Resolution IV or More," *Journal of Statistical Planning and Inference*, 137, 317–323. [219]

Caporossi, G., and Hansen, P. (2004), "Variable Neighborhood Search for Extremal Graphs. 5. Three Ways to Automate Finding Conjectures," *Discrete Mathematics*, 276, 81–94. [222]

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. [220,225,227,229,230]

Cheng, C.-S., Mee, R. W., and Yee, O. (2008), "Second Order Saturated Orthogonal Arrays of Strength Three," *Statistica Sinica*, 18, 105–119. [219,220,221,227,229]

Deng, L.-Y., and Tang, B. (1999), "Generalized Resolution and Minimum Aberration Criteria for Plackett-Burman and Other Nonregular Factorial Designs," *Statistica Sinica*, 9, 1071–1082. [220]

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, 1–28. [230]

Eglese, R. (1990), "Simulated Annealing: A Tool for Operational Research," *European Journal of Operational Research*, 46, 271–281. [222]

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. [222]

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. [222]

Glover, F., and Laguna, M. (1997), *Tabu Search*, New York: Springer. [222]

Hansen, P., and Mladenović, N. (2001), "Variable Neighborhood Search: Principles and Applications," *European Journal of Operational Research*, 130, 449–467. [222]

Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2008), "Variable Neighbourhood Search: Methods and Applications," *4OR*, 6, 319–360. [222]

Hedayat, A., Sloane, N., and Stufken, J. (1999), *Orthogonal Arrays: Theory and Applications*, New York: Springer. [219,228]

Houston, D. X., Ferreira, S., Collofello, J. S., Montgomery, D. C., Mackulak, G. T., and Shunk, D. L. (2001), "Behavioral Characterization: Finding and Using the Influential Factors in Software Process Simulation Models," *The Journal of Systems and Software*, 59, 259–270. [219,230]

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. [222]

Li, W., and Lin, D. K. J. (2003), "Optimal Foldover Plans for Two-Level Fractional Factorial Designs," *Technometrics*, 45, 142–149. [220]

——— (2016), "A Note on Foldover of 2^{k-p} Designs With Column Permutations," *Technometrics*, 58, 508–512. [220,223,224]

Li, W., Lin, D. K. J., and Ye, K. Q. (2003), "Optimal Foldover Plans for Two-Level Nonregular Orthogonal Designs," *Technometrics*, 45, 347–351. [220]

Li, W., and Nachtsheim, C. (2000), "Model-Robust Factorial Designs," *Technometrics*, 42, 345–352. [230]

Marley, C. J., and Woods, D. C. (2010), "A Comparison of Design and Model Selection Methods for Supersaturated Experiments," *Computational Statistics and Data Analysis*, 54, 3158–3167. [229,230]

Mee, R. (2009), *A Comprehensive Guide to Factorial Two-Level Experimentation*, New York: Springer. [219]

Mee, R. W., Schoen, E. D., and Edwards, D. J. (2017), "Selecting an Orthogonal or Non-Orthogonal Two-Level Design for Screening," *Technometrics*, 59, 305–318. [219]

Michalewicz, Z., and Fogel, D. (2004), *How to Solve It: Modern Heuristics*, New York: Springer. [221]

Miller, A., and Sitter, R. R. (2001), "Using the Folded-Over 12-Run Plackett-Burman Design to Consider Interactions," *Technometrics*, 43, 44–55. [230]

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. [222]

Sartono, B., Goos, P., and Schoen, E. (2015), "Constructing General Orthogonal Fractional Factorial Split-Plot Designs," *Technometrics*, 57, 488–502. [222]

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. [220,225,226,228,229]

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. [222]

Sloane, N. J. A. (1999), "A Library of Hadamard Matrices," available at *http://www.neilsloane.com/Hadamard/* [225]

Smucker, B., del Castillo, E., and Rosenberg, J. (2011), "Exchange Algorithms for Constructing Model-Robust Experimental Designs," *Journal of Quality Technology*, 43, 28–42. [230]

Smucker, B., del Castillo, E., and Rosenberg, J. (2012), "Model-Robust Two-Level Designs Using Coordinate Exchange Algorithms and a Maximin Criterion," *Technometrics*, 54, 367–375. [230]

Smucker, B., and Drew, N. (2015), "Approximate Model Spaces for Model-Robust Experiment Design," *Technometrics*, 57, 54–63. [230]

Sun, D. X., Li, W., and Ye, K. Q. (2008), "Algorithmic Construction of Catalogs of Non-Isomorphic Two-Level Orthogonal Designs for Economic Run Sizes," *Statistics and Application*, 6, 144–158. [219]

Sun, D. X., Wu, C. F. J., and Chen, Y. (1997), "Optimal Blocking Schemes for 2^n and 2^{n-p} Designs," *Technometrics*, 39, 298–307. [226]

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. [222]

Tang, B., and Deng, L.-Y. (1999), "Minimum G_2-Aberration for Nonregular Fractional Factorial Designs," *The Annals of Statistics*, 27, 1914–1926. [220]

Wu, C. F. J., and Hamada, M. S. (2009), *Experiments: Planning, Analysis, and Optimization*, New York: Wiley. [219]

Xu, H. (2005), "Some Nonregular Designs From the Nordstrom-Robinson Code and Their Statistical Properties," *Biometrika*, 92, 385–397. [227]

——— (2009), "Algorithmic Construction of Efficient Fractional Factorial Designs With Large Run Sizes," *Technometrics*, 51, 262–277. [220,230]

Xu, H., and Wong, A. (2007), "Two-Level Nonregular Designs From Quaternary Linear Codes," *Statistica Sinica*, 17, 1191–1213. [220,225,227,229,230,231]

Supplementary sections to

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

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

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

April 30, 2018

This document includes the following sections.

A Objective functions and fast update methods

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

A.1 B_4 optimization

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The number of calculations required by the updating formulas for matrix C is clearly smaller than the matrix multiplication $D_u D_l^T$ when $m > 5$. More importantly, the number of calculations required by the updating formulas does not depend on the numbers of factors in the concatenated design. As a specific example, for two parent designs with 32 runs and 10 factors, the number of calculations required for a complete update of C when

making a change in D_l is 19,456; the number of calculations required by the quick updating procedure is 2,080 for a sign switch of a column, and 8,256 for a swap between two columns. The difference between the number of calculations increases with the number of factors. Although the number of calculations saved by either updating formula suggests that both should be included in the implementation of our CC algorithm, the Matlab implementation only includes the updating formula for a sign switch. A computing time study (not shown) revealed that the updating formula for a swap of two columns requires the same or slightly more time than just changing the positions of the two columns and calculating matrix C from scratch. This is probably due to computer memory allocation.

A.2 F_4 optimization

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

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

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

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

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

Now, consider the matrix

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

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

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

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

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

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

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

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

B Algorithm performance evaluation

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

B.1 Design cases

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

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

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

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

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

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

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

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

B.2 CC algorithm

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

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

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

To evaluate the F_4 optimization, we check the generalized resolution as well as the $f(D)$ values of the concatenated designs. Figure 2 shows the distribution of the generalized resolutions of the concatenated designs from the random search and the CC algorithm. Figure 2a shows that, for the OA64One case, the CC algorithm produces substantially more designs with a generalized resolution of 4.5 than the random search. The fact that this resolution is reached in only 10% of the runs of the algorithm suggests that at least 10 restarts of the CC algorithm are needed for an optimal result with the stand alone CC algorithm. Regarding the OA64two case, Figure 2b shows that the CC algorithm and the random search resulted in the same number of concatenated designs with generalized resolution 4.25 and with generalized resolution 4.5. For the 80-run case, Figure 2c shows that the CC algorithm only produced concatenated designs with a generalized resolution of 4.6, while the random search generated 85 designs with a generalized resolution of 4.4. For the OA96One case, the CC algorithm created 105 concatenated designs with a generalized resolution of 4.5, whereas the random search produced only designs with a generalized resolution of 4.33; see Figure 2d. Finally, all concatenated designs for the OA96Two case

had a resolution of 4.67, regardless of whether the CC algorithm or the random search was used. For this reason, Figure 2 does not include a separate panel for the OA96Two case.

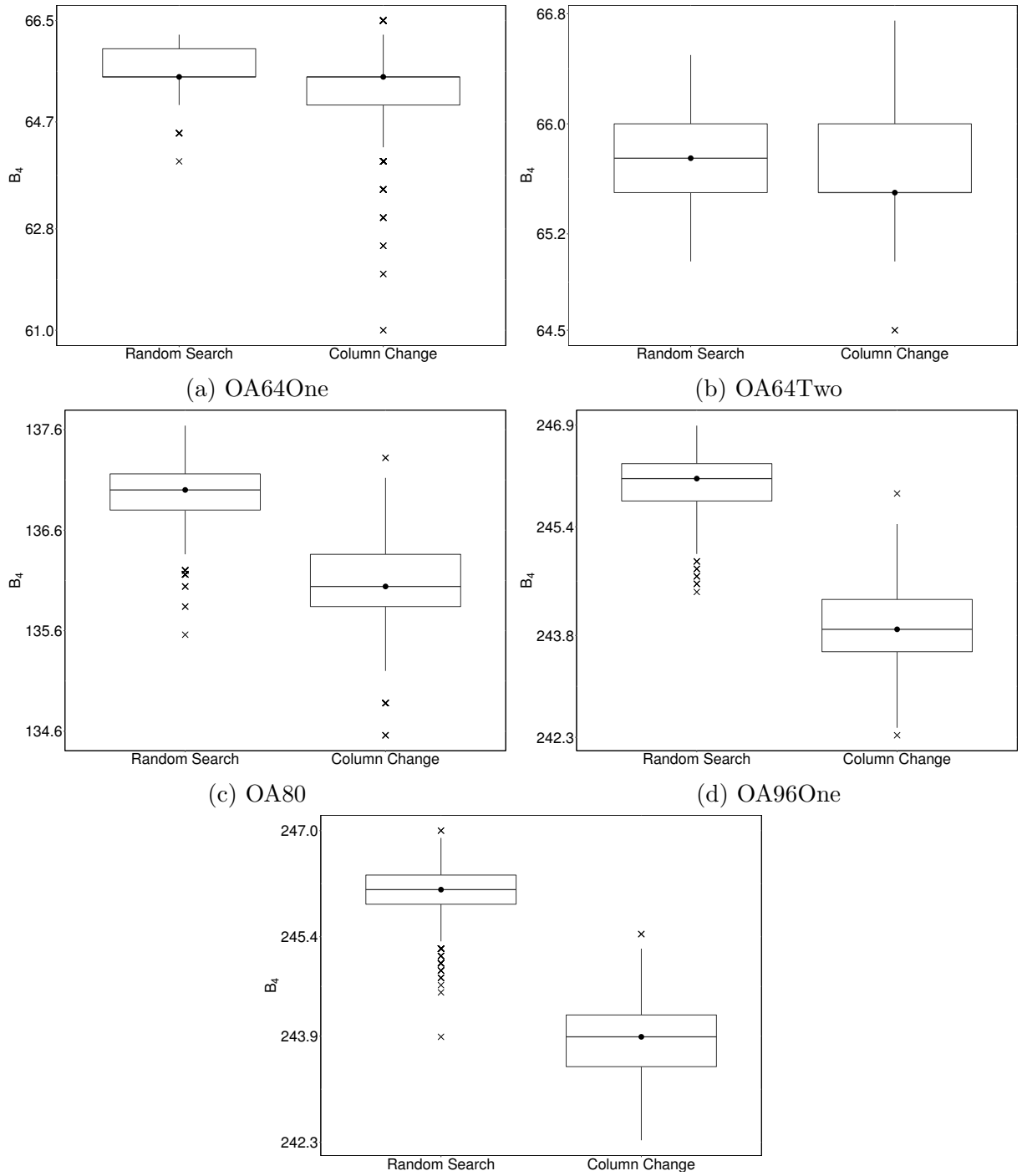
We now turn to the $f(D)$ value of the designs that optimize the F_4 vector. Recall from Section A.2 that $f(D)$ is a linear combination of the elements of the F_4 vector, in which the frequencies of large J_4-characteristics receive a larger weight than those of small J_4-characteristics. Low values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

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

B.3 Neighborhood structures

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

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



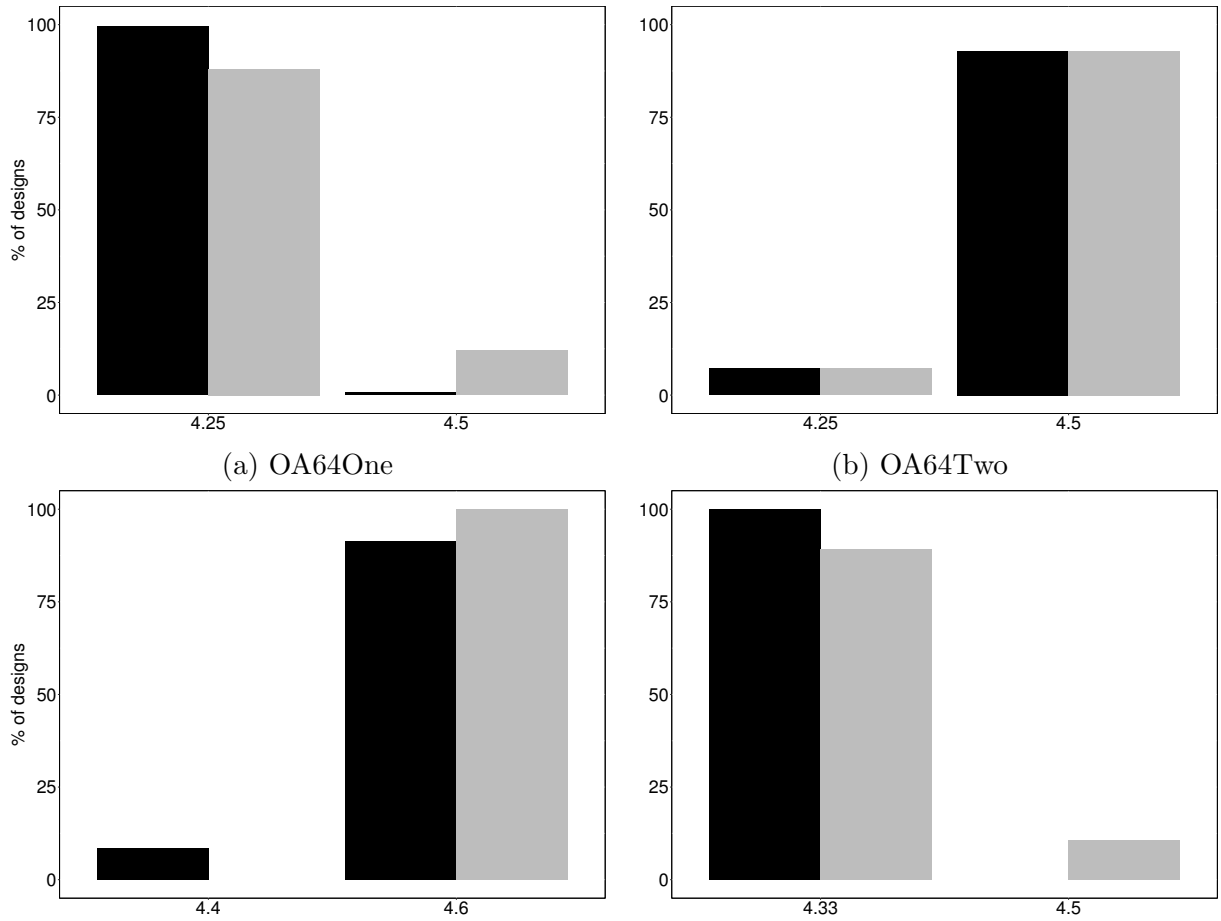
(a) OA64One (b) OA64Two

(c) OA80 (d) OA96One

(e) OA96Two

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

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



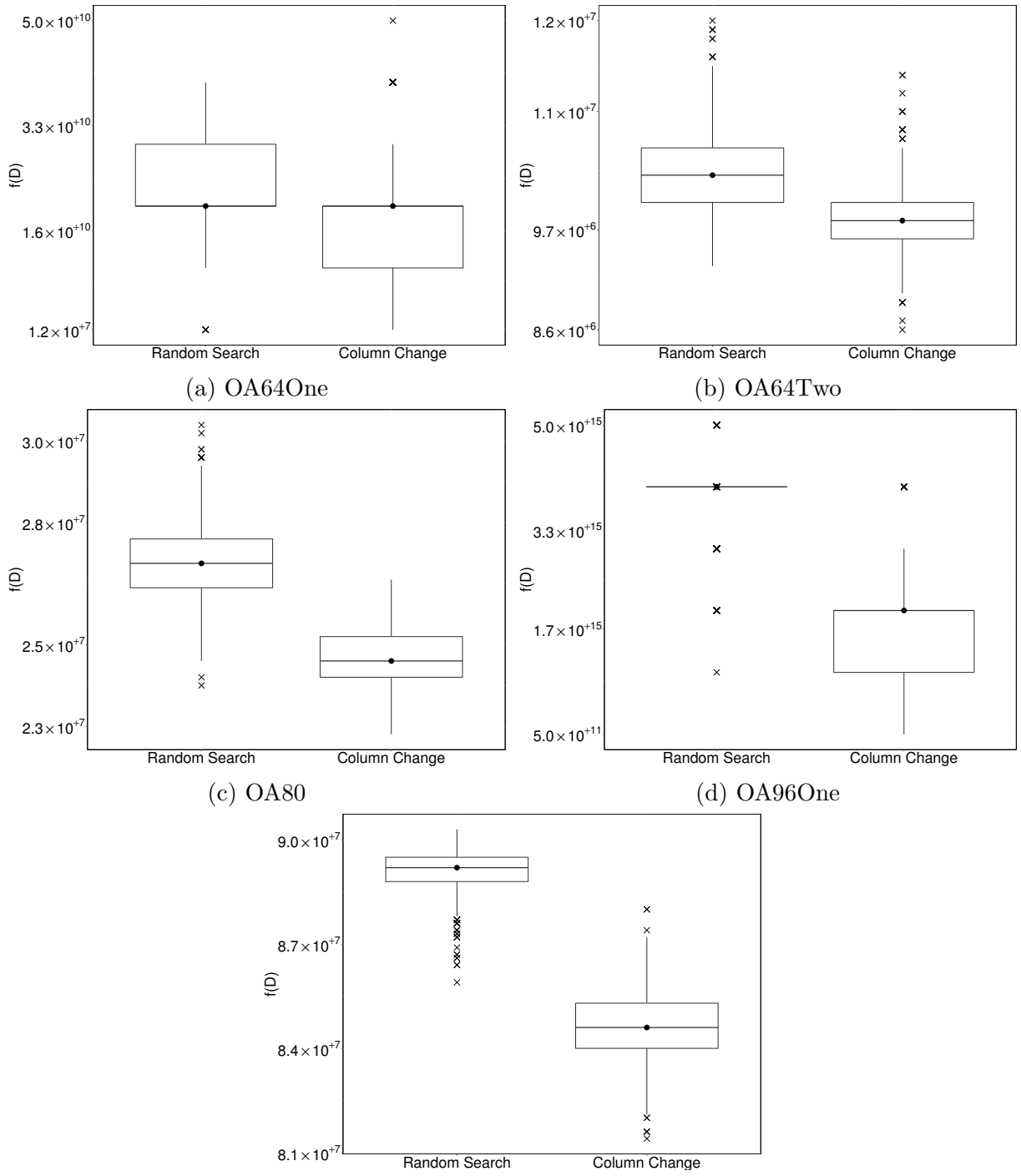
(a) OA64One (b) OA64Two

(c) OA80 (d) OA96One

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

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

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



(a) OA64One

(b) OA64Two

(c) OA80

(d) OA96One

(e) OA96Two

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

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

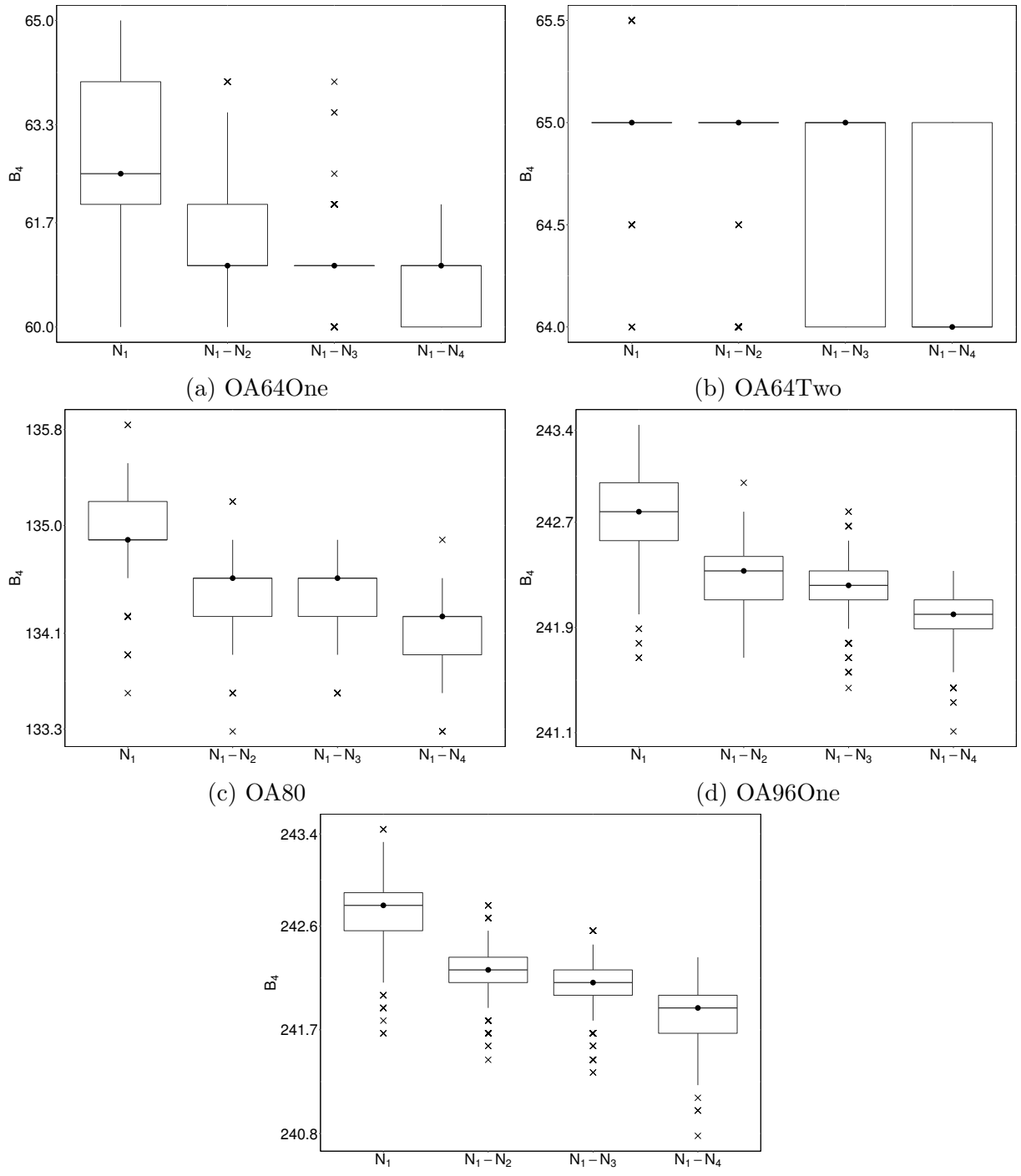
B.4 Performance for 128-run designs

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

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

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

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



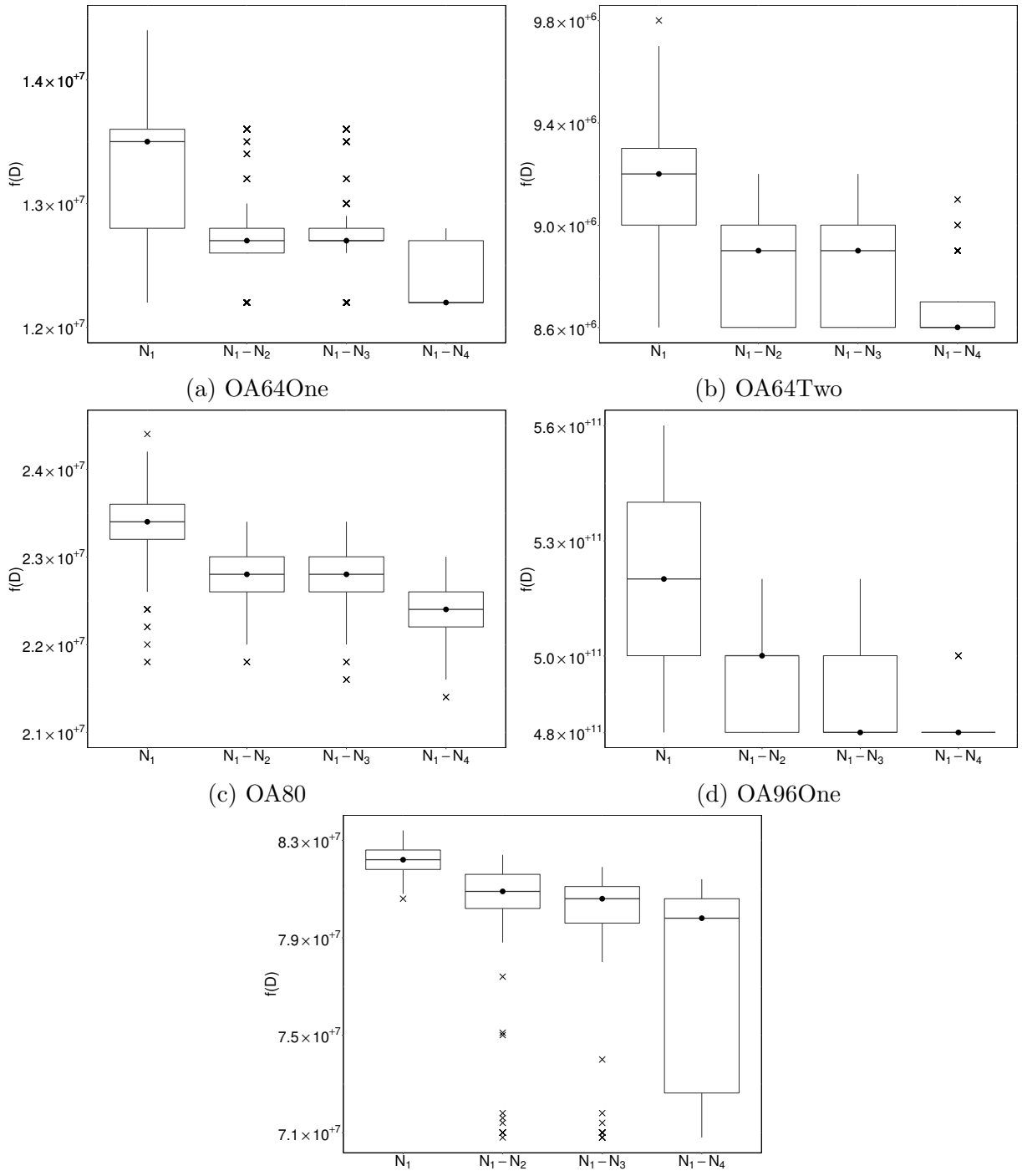
(a) OA64One (b) OA64Two

(c) OA80 (d) OA96One

(e) OA96Two

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

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



(a) OA64One (b) OA64Two

(c) OA80 (d) OA96One

(e) OA96Two

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

designs.

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

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

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

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

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

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

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

C Parent designs

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

at random.

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

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

D Tables of concatenated designs

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

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

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

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

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

For 64 runs, the concatenated designs based on different parent designs are 9.b, 10.b,

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

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

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

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

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

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

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

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

E 128-run designs of strength 4

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

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

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

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

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

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

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

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

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

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

References

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

Chen, J., Sun, D. X., and Wu, C. F. J. (1993). A catalogue of two-level and three-level fractional factorial designs with small runs. *International Statistical Review*, 61:131–145.

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

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

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

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

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

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