Systematic enumeration of two-level even-odd designs of strength 3

Statistics and Computing manuscript No.
(will be inserted by the editor)

Systematic Enumeration of Definitive Screening Designs

Eric D. Schoen · Pieter T. Eendebak · Alan Vazquez · Peter Goos

Received: date / Accepted: date

Abstract Conference designs are $n \times k$ matrices, $k \leq n$, with orthogonal columns, one zero in each column, at most one zero in each row, and -1 and $+1$ entries elsewhere. Conference designs with $k = n$ are called conference matrices. Definitive screening designs (DSDs) are constructed by folding over a conference design and adding a row vector of zeros. We propose methodology for the systematic enumeration of conference designs with a specified number of rows and columns, and thereby for the systematic enumeration of the corresponding DSDs. We demonstrate its potential by enumerating all conference designs with up to 24 rows and columns, and thus all DSDs with up to 49 runs. A large fraction of these DSDs cannot be obtained from conference matrices and is therefore new to the literature. We identify DSDs that minimize the correlation among contrast vectors of second-order effects and provide them in supplementary files.

Keywords Conference Matrix · Generalized Aberration · Isomorphism Class · Sequential Enumeration

1 Introduction

A key step in optimizing processes or developing new products is to identify, from a list of candidate factors, those that really affect the process or product properties. This step is called factor screening, and a vast body of statistical design literature is devoted to the construction of efficient plans to approach factor screening experimentally. Within this vast body of literature, two-level screening designs have received most of the attention. The experimental results from a two-level screening design usually permit identification of substantial linear effects and, depending on the design, a few two-factor interactions; see Schoen et al. (2017) and Mee et al. (2017) for recent reviews.

When all the factors are quantitative, it is of practical interest to check for quadratic effects as well. With this purpose in mind, Jones and Nachtsheim (2011) developed definitive screening designs (DSDs) for experiments involving quantitative factors. These designs study all factors at three equidistant levels to estimate quadratic effects in addition to linear effects and two-factor interaction effects. Since their introduction in the literature, both papers on further developments of the DSDs and on applications of these designs have appeared. Papers of the former kind include Xiao et al. (2012), Jones and Nachtsheim (2013), Georgiou et al. (2014), Nguyen and Pham (2016), Jones and Nachtsheim (2017), Nachtsheim et al. (2017), Schoen et al. (2019) and Vazquez et al. (2020). Papers of the latter kind include Olsen et al. (2014), Renzi et al. (2014), Libbrecht et al. (2015), Tai et al. (2015), Dougherty et al. (2015), Fidaleo et al. (2016), Maestroni et al. (2018), and Jones and Lanzerath (2021).

Following Xiao et al. (2012), the DSDs studied in the current literature are all based on conference matrices; see also Nguyen and Stylianou (2013) and Phoa and Lin (2015). A conference matrix $\mathbf{C}$ of order n is an $n \times n$ matrix with elements $c_{ij} \in \{-1, 0, 1\}$ such that $\mathbf{C}^T\mathbf{C} = (n-1)\mathbf{I}_n$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix.

This research was financially supported by Fonds Wetenschappelijk Onderzoek (FWO, Flanders, Belgium).

Eric D. Schoen[1] E-mail: eric.schoen@kuleuven.be · Pieter T. Eendebak[2] E-mail: pieter.eendebak@gmail.com · Alan R. Vazquez[3] E-mail: alanv@uark.edu · Peter Goos[1,2] E-mail: peter.goos@kuleuven.be
[1]KU Leuven, Faculty of Bioscience Engineering
[2]University of Antwerp, Faculty of Business and Economics
[3]University of Arkansas, Department of Industrial Engineering

The definition implies that every row and column of a conference matrix has one zero entry. A standard DSD is constructed by folding over a conference matrix $\mathbf{C}$ and adding a row vector of zeros. So, the structure of a standard DSD is given by $[\mathbf{C}^T, -\mathbf{C}^T, \mathbf{0}_n]^T$, where $\mathbf{0}_n$ is an n-dimensional column vector of zeros. A standard DSD therefore involves $N = 2n + 1$ runs and n factors. As the columns of a conference matrix are orthogonal, a standard DSD has contrast vectors for linear effects that are orthogonal to each other. Due to the folding over, the contrast vectors for the linear effects are also orthogonal to those of the second-order effects.

Conference matrices do not exist when n is odd, and when n is 22, 34 or 58 (Colbourn and Dinitz, 2006). Therefore, one cannot construct standard DSDs with $N = 2n + 1$ runs for odd values of n or when $n \in \{22, 34, 58\}$. To deal with this problem, Xiao et al. (2012) recommended dropping columns from standard DSDs with 1–3 columns more than the required number. The best sets of columns to drop from standard DSDs with up to 24 factors are presented by Vazquez et al. (2020).

The DSDs which are available in statistical software and studied in the literature are all based on the conference matrices given by Xiao et al. (2012). However, for orders $n \geq 20$, there are several conference matrices to construct a DSD from (Greig et al., 2006). In addition, there may be DSDs that cannot be obtained from a conference matrix. Therefore, it would be useful to collect alternative DSDs and evaluate them according to appropriate statistical criteria. Of particular interest are criteria based on correlations among two-factor interaction contrast vectors, as it is known that DSDs with the same run size may differ in these correlations (Schoen et al., 2019).

Our approach to construct DSDs builds on the notion of conference designs. A conference design is similar to a conference matrix, but it can have fewer columns. So, a conference design $\mathbf{X}$ is an $n \times k$ matrix, with elements $x_{ij} \in \{-1, 0, 1\}$, columns $x_1, \ldots, x_k$, $k \leq n$, one 0 in every column and at most one 0 in each row, such that $\mathbf{X}^T\mathbf{X} = (n-1)\mathbf{I}_k$. A conference matrix is thus a conference design for which $k = n$. We use the name DSD for any design constructed from an $n \times k$ conference design $\mathbf{X}$ by folding it over and adding a zero row. So, the DSDs we discuss have the form $[\mathbf{X}^T, -\mathbf{X}^T, \mathbf{0}_k]^T$ and include orthogonal contrast vectors for the linear effects. We refer to n as the row size of the conference design, to $N = 2n + 1$ as the run size of a DSD and to k as the column size of the conference design or the number of factors of the corresponding DSD. Due to the folding-over construction, the first-order effects of the factors in a DSD are orthogonal to the second-order effects.

The goal of this paper is to present methodology for the systematic enumeration of conference designs and their corresponding DSDs, and to investigate the existence of DSDs that cannot be constructed from conference matrices. To this end, we introduce a novel enumeration algorithm, called the minimum complete set (MCS) algorithm. Two features of this algorithm are particularly attractive. First, it is computationally inexpensive, so that it can generate complete catalogs of DSDs for most practical applications. For instance, it can generate DSDs with up to 20 factors and 41 runs in a matter of seconds. Second, since all DSDs of a specific run size are produced, the algorithm guarantees that the best DSD is obtained in terms of any criterion.

The rest of this paper is organized as follows. In Section 2, we define equivalence of conference designs and the corresponding definitive screening designs, and describe a way of ordering the designs in a given equivalence class. We show that it suffices to retain only one design of each equivalence class, and propose a convenient form of such a design. Building on these notions, Section 3 presents our MCS algorithm. In Section 4, we demonstrate the potential of the algorithm by enumerating minimum complete sets of all conference designs with up to 24 rows and identifying the enumerated DSDs that minimize two design criteria based on the correlation among two-factor interaction contrast vectors. Finally, we discuss the strengths and weaknesses of our approach in Section 5. Files with designs that minimize the correlation-based criteria are available in the supplementary materials to this paper.

2 Isomorphism of conference designs and definitive screening designs

For a given row size n and column size k, there can be many conference designs. Two conference designs are isomorphic if one can be obtained from the other by permuting rows or columns, or using sign switches within one or more columns. The set of isomorphic conference designs is called an isomorphism class (Schoen et al., 2019). The designs in such a class result in isomorphic DSDs. In these DSDs, the absolute correlations between any two contrast vectors modeling main effects or second-order effects are the same. As a result, for any model with an intercept, some linear effects and some second-order effects collected in model matrices $\mathbf{K}$, isomorphic DSDs have the same information matrices $\mathbf{K}^T\mathbf{K}$ up to the signs of cross-products involving sign-switched columns. As almost all current evaluation criteria for screening designs are based on absolute correlations among contrast vectors or the information matrix, these criteria are the same for the DSDs from

a given isomorphism class. These criteria include combinatorial criteria such as β-aberration (Cheng and Ye, 2004) or G-aberration (Schoen et al., 2019), which summarize correlations among contrast vectors, and model-based criteria such as D- or A-optimality (Atkinson et al., 2007), which account for the determinant of the information matrix and the trace of its inverse, respectively.

From a statistical point of view, it is interesting to identify conference designs that do not belong to the same isomorphism class, because the corresponding DSDs may differ in the values of one or more evaluation criteria. This is why we need an algorithm to enumerate one representative from each isomorphism class.

Unlike in the case of orthogonal arrays (Schoen et al., 2010), we consider sign switches of rows in conference designs to be isomorphism operations, because a DSD constructed from a conference design includes the original conference design and its mirror image. A sign switch in any row of a design results in a mirror image of that row that is equal to the original. The net result is that the same DSD is obtained regardless of whether the original row is included in the conference design or its mirror image. So, because of our focus on DSDs, we need to consider only one version of each row when enumerating conference designs. To the best of our knowledge, this is the first time that a sign switch in rows is considered to be an isomorphism operation for statistical designs.

For the enumeration methodology for conference designs that we develop in Section 3, we define a representative form of each isomorphism class that is easy to check. We call this form the lexicographically maximal with zero, or LM0, representative. For its definition, we introduce the L0 ordering, a lexicographic ordering of the elements of a column, entire columns and entire conference designs, with a special role for the zeros.

Definition 1 The L0 order of the factor levels 1 and -1 is $1 \succ -1$, where $\succ$ denotes 'is greater than'.

Columns where the zero appears in an earlier position are defined to be larger than those where the zero appears in a later position. The L0 ordering of columns is defined by both the position of the zero and the L0 ordering of the other elements:

Definition 2 A column $\mathbf{a}$ is larger than a column $\mathbf{b}$ in the L0 ordering, which is denoted by $\mathbf{a} \succ \mathbf{b}$, if either of the following conditions hold:

1. The zero in column $\mathbf{a}$ appears in an earlier position than the zero in column $\mathbf{b}$.
2. The zero in column $\mathbf{a}$ appears in the same position as the zero in column $\mathbf{b}$, and the first element in which the columns differ is 1 in $\mathbf{a}$ and -1 in $\mathbf{b}$.

Table 1 Isomorphic conference designs with 8 rows and 3 columns

Design 1			Design 2			Design 3		
0	1	1	0	1	1	0	1	1
1	0	−1	1	0	1	1	0	1
1	1	0	1	1	1	1	1	1
1	1	1	1	1	−1	1	1	−1
1	1	−1	1	1	−1	1	1	−1
1	−1	1	1	−1	0	1	−1	0
1	−1	1	1	−1	1	1	−1	−1
1	−1	−1	1	−1	−1	1	−1	1

The ordering of entire conference designs is derived from the ordering of the columns:

Definition 3 Conference design $\mathbf{A}$ is larger than conference design $\mathbf{B}$ in the L0 ordering, which is denoted by $\mathbf{A} \succ \mathbf{B}$, if the first column where the designs differ is larger in $\mathbf{A}$ than in $\mathbf{B}$.

Finally, based on the L0 ordering, we define the LM0 representative of an isomorphism class as follows:

Definition 4 The LM0 representative of an isomorphism class of conference designs with n rows and k columns is the largest design within its isomorphism class under the L0 ordering.

To illustrate these four definitions, consider the three conference designs in Table 1. The designs only differ in their third columns. Therefore, by Definition 3, the L0 ordering of the three designs is completely determined by these columns. Design 1 is the largest design of the three, because the zero in its third column appears in an earlier position than the zeros in the third columns of the other two designs (see Definition 2). Designs 2 and 3 have the zero in the same position in their last column, and the first row in which the designs' last columns differ is the seventh. The symbol in that position is a 1 for design 2 and a -1 for design 3. As, by Definition 1, $1 \succ -1$ in the L0 ordering, the third column of design 2 is larger than the third column of design 3 (see Definition 2). Therefore, design 2 is larger than design 3 according to the L0 ordering.

All three designs in Table 1 are isomorphic. Design 2 can be converted into design 1 by swapping the second and third columns, sorting the rows so that the zeros are on the diagonal, and sorting the remaining rows to maximize the design according to the L0 ordering. Design 3 can be turned into design 2 by swapping the seventh and eighth rows. Finally, to illustrate Definition 4, design 1 is in fact the largest design within its isomorphism class, and therefore the LM0 representative of its class.

Table 2 Some 16-row conference designs with k columns. Columns x_1, x_2, x_3: LM0 design for $k = 3$; columns x_1, x_2, x_{3*}: alternative design for $k = 3$; columns x_1, x_2, x_3 with one of the columns $x_{4a} - x_{4d}$: four different LM0 designs for $k = 4$.

Row	x_1	x_2	x_3	x_{3*}	x_{4a}	x_{4b}	x_{4c}	x_{4d}
1	0	1	1	1	1	1	1	1
2	1	0	−1	1	−1	−1	1	1
3	1	1	0	1	−1	−1	−1	−1
4	1	1	1	1	0	0	1	1
5	1	1	1	1	1	1	1	1
6	1	1	1	−1	1	−1	1	−1
7	1	1	−1	−1	1	1	−1	1
8	1	1	−1	−1	−1	1	−1	−1
9	1	1	−1	−1	−1	−1	−1	−1
10	1	−1	1	0	1	1	0	0
11	1	−1	1	1	−1	1	−1	1
12	1	−1	1	1	−1	−1	−1	−1
13	1	−1	1	1	−1	−1	−1	−1
14	1	−1	−1	−1	1	1	1	1
15	1	−1	−1	−1	1	1	1	1
16	1	−1	−1	−1	1	−1	1	−1

Algorithm 1: Pseudocode of the minimum complete set (MCS) algorithm.

Input: Number of rows n

1 $B \leftarrow C^*(n, 3)$ `/* unique LM0 design with` n `rows and 3 columns */`

2 Determine $s(B, 2)$ `/* row permutations preserving first two columns of` B `*/`

3 Determine $G(B, 3)$ `/* two options for third column of` B `orthogonal to the first two columns and maximal with respect to` $s(B, 2)$ `*/`

4 **for** $k = 3, \ldots, n-1$ **do**

5 $\quad C^+(n, k+1) \leftarrow \varnothing$

6 $\quad$ **forall** $D \in C^*(n, k)$ **do**

7 $\quad\quad s(D, k), G(D, k+1), E^+(D) \leftarrow$ `extension`$(D, s(D, k-1), G(D, k))$ `/* row permutations preserving all` k `columns of` D`, orthogonal and maximal options for column` $k+1$`, set of designs with` $k+1$ `columns */`

8 $\quad\quad C^+(n, k+1) \leftarrow C^+(n, k+1) \cup E^+(D)$

9 $\quad C^*(n, k+1) \leftarrow$ `reduction`$(C^+(n, k+1))$

Output: Minimum complete sets of LM0 conference designs $C^*(n, 4), \ldots, C^*(n, n)$

3 Enumeration methodology

In this section, we present a systematic way to enumerate conference designs. For this purpose, we developed a suite of algorithms. The main algorithm is called the minimum complete set (MCS) algorithm. Its purpose is to create a minimum complete set of conference designs for a given number of rows and columns. Such a set is minimum complete because it includes exactly one representative of each isomorphism class. The MCS algorithm calls two subsidiary algorithms, which are named the extension algorithm and the reduction algorithm. The extension algorithm takes each design from a minimum complete set of conference designs with n rows and k columns as input and returns conference designs with n rows and $k+1$ columns. Together, these extended designs form a complete set, because it includes at least one representative of each isomorphism class. The reduction algorithm reduces that set to a minimum complete set. We present the MCS, extension and reduction algorithms in Sections 3.1, 3.2 and 3.3, respectively. An open-source implementation of our MCS algorithm and subsidiary algorithms is provided by Eendebak and Vazquez (2019).

We exemplify the algorithms by constructing all LM0 conference designs with 16 rows and 4 columns from the single 16-row 3-column LM0 conference design. Table 2 is a reference table for these designs. The 3-column LM0 design includes the columns labeled x_1, x_2 and x_3. An alternative 3-column design that is not of LM0 form includes the columns labeled x_1, x_2 and x_{3*}. The four 4-column LM0 designs include the 3-column LM0 design along with one of the columns labeled x_{4a}, x_{4b}, x_{4c} or x_{4d}.

3.1 MCS algorithm

Pseudocode for the MCS algorithm is shown in Algorithm 1. The algorithm uses the number of rows n as input. That number also defines the maximum number of columns of the conference designs to be generated. The MCS algorithm produces minimum complete sets of conference designs in LM0 form with the specified row size n and k columns, for k ranging from 4 to n. The sets are denoted by $C^*(n, k)$.

Schoen et al. (2019) showed that the set $C^*(n, 3)$ includes only one design. The MCS algorithm starts from this design, which is designated B in line 1 of Algorithm 1. The next step, shown in line 2, is to determine the group of row permutations that preserve the first two columns of the starting design B. We call this group

the row symmetry group and denote it by $s(B, 2)$. For the 16-row LM0 design involving the columns x_1, x_2 and x_3 in Table 2, this symmetry group consists of $(7!)^2$ row permutations, because there are 7! row permutations that do not affect the first two columns' elements in rows 3–9 and 7! row permutations that do not affect the first two columns' elements in rows 10–16. Any row permutation involving at least one of the first two rows of B alters the first two columns. Therefore, no such permutation is included in $s(B, 2)$.

The third step of the MCS algorithm, shown in line 3, calculates all options for the third column that are orthogonal to the first two columns and maximal with respect to $s(B, 2)$. In other words, the algorithm determines all columns similar to the third column that cannot be made larger in terms of the L0 ordering by any permutation in $s(B, 2)$. Schoen et al. (2019) showed that there are two such columns for any given value of n. We denote the set of options for the third column by $G(B, 3)$. For the 16-row LM0 design, this set consists of the columns x_3 and x_{3*} in Table 2.

The set of row permutations in $s(B, 2)$ and the set of columns in $G(B, 3)$ are used in the extension algorithm to construct all possible columns that are orthogonal to the first two columns. These columns are subjected to further tests to determine whether they are suitable candidates for the fourth column of the design.

After the initialization, the algorithm proceeds cycle by cycle (see line 4). Each cycle consists of an extension part followed by a reduction part. In the extension part, complete sets of conference designs with one extra column are generated such that the LM0 representatives of each isomorphism class are always included. In the reduction part, the extended designs that are not in LM0 form are removed.

In line 5 of Algorithm 1, the set $C^+(n, k+1)$, which will contain the candidate extended designs for a new cycle, is initialized. Line 6 is a loop over all designs D with k columns that have to be extended with an extra column.

In line 7, the extension algorithm is called. It uses (i) the current design D, (ii) the row symmetry group $s(D, k-1)$, and (iii) the options for the kth column in $G(D, k)$ as inputs. The extension algorithm uses these options to calculate candidates for column $k+1$. The output of the extension algorithm consists of three sets. First, the row symmetry group of the entire design D, $s(D, k)$, is calculated. That group is needed in the next extension cycle. The second set, $G(D, k+1)$, involves the options for column $k+1$ that are maximal with respect to $s(D, k)$ and orthogonal to all columns of D. The third set, $E^+(D)$, consists of extended designs formed by appending each of the options in $G(D, k+1)$ to D.

Returning to the 16-row example, the symmetry group $s(B, 3)$ of the entire design B is smaller than $s(B, 2)$, because it involves row permutations that must leave three columns unchanged rather than two. In particular, the group includes only $(3!)^3(4!)$ row permutations. Further, recall that $G(B, 3)$ includes the columns x_3 and x_{3*} in Table 2. Starting from $G(B, 3)$, the extension algorithm identifies 13 candidate columns to extend B, six candidate columns based on x_3 and seven candidate columns based on x_{3*}. These columns are all maximal with respect to $s(B, 3)$ and orthogonal to the third column of B, namely x_3. We refer to the set of orthogonal and maximal candidate extensions as $G(B, 4)$. Appending these candidates to B, we obtain a set of 13 4-column conference designs, $E^+(B)$. As B is the only design in $C^*(16, 3)$, the set $C^+(16, 4)$ of Algorithm 1 includes only the 13 designs of $E^+(B)$.

The extended designs obtained from a given design D are added to the set $C^+(n, k+1)$ in line 8 of Algorithm 1. In the final line of the MCS algorithm, a call to the reduction algorithm reduces the set $C^+(n, k+1)$ by discarding the designs that are not in LM0 form.

The end result of the first cycle in the 16-row design case is a set of four designs that form $C^*(16, 4)$. So, the reduction algorithm removes 9 of the 13 candidate designs. The four remaining designs are shown in Table 2: they all involve the columns x_1, x_2 and x_3, because we seek to extend design B formed by these columns, and one of the columns $x_{4a} - x_{4d}$. The fact that none of the LM0 designs in $C^*(16, 4)$ have x_{3*} as their fourth column could be interpreted as evidence that the intermediate step in which x_{3*} is created can be skipped. However, the x_{4c} and x_{4d} columns of two 4-column LM0 representatives in Table 2 can only be found by considering the three-column design with x_{3*} as its third column. Creating x_{3*} is thus a necessary intermediate step in the enumeration of all isomorphism classes.

3.2 Extension algorithm

A brute force extension method of a design with n rows and k columns would generate all possible columns with one zero, $n/2$ elements equaling $+1$ and $n/2-1$ elements equaling -1. We use a more efficient approach instead by only considering columns that are smaller than and orthogonal to the first $k-1$ columns. Pseudocode of the extension algorithm is shown in Algorithm 2. The extension algorithm's inputs are (i) an LM0 design D with n rows and k columns, (ii) the row symmetry group $s(D, k-1)$, containing row permutations that preserve

the first $k-1$ columns of D, and (iii) the set $G(D,k)$ that includes all options for the kth column of D orthogonal to the first $k-1$ columns and maximal with respect to the row symmetry group $s(D,k-1)$. In lines 1–3 of Algorithm 2, the row symmetry group of the entire design D, $s(D,k)$, is calculated, and the output sets $G(D,k+1)$ and $E^+(D)$ are initialized.

Next, all alternatives for the $(k+1)$st column of the input design are generated by applying the permutations of the row symmetry group $s(D,k-1)$ to each option in $G(D,k)$. The alternatives are collected in the set $G^-(D,k+1)$; see line 5. All columns in the set $G^-(D,k+1)$ are orthogonal to the columns 1 up to $k-1$, because they differ only by a row permutation that preserves the initial $k-1$ columns.

In line 6, the alternatives in the set $G^-(D,k+1)$ that are not maximal with respect to the symmetry group $s(D,k)$ are discarded. This is because these alternatives will not lead to a design that is maximal according to the L0 ordering. All columns in the updated set $G^-(D,k+1)$ are orthogonal to the columns 1 up to $k-1$ but not necessarily orthogonal to the original kth column of the input design D. Therefore, an orthogonality test for each column in $G^-(D,k+1)$ is performed, and only columns that are orthogonal to the original kth column of the input design D are retained (line 7). The columns that pass the orthogonality test are then added to $G(D,k+1)$ (see line 8). Finally, the set of extended designs with $k+1$ columns, $E^+(D)$, is built by appending each column in $G(D,k+1)$ to the input design D; see lines 10 and 11 of Algorithm 2. Since not all designs in $E^+(D$ are in LM0 form, we need to discard the designs that are not in that form. This is done in the reduction part of the MCS algorithm.

Returning to the 16-row example in Table 2, every candidate for the fourth column must both be smaller than the first three columns and orthogonal to these columns. Line 5 of the extension algorithm applies all $(7!)^2$ row permutations in $s(B,2)$ to each of the two columns x_3 and x_{3*} in $G(B,3)$ and collects the results in the set $G^-(B,4)$. At this stage, the set $G^-(B,4)$ includes all columns smaller than and orthogonal to the first two columns, including columns x_3 and x_{3*}. The set is complete in the sense that L0 representatives for $k=4$ must involve one of the columns in $G^-(B,4)$.

In line 6 of the extension algorithm, only the columns of $G^-(B,4)$ that are maximal with respect to $s(B,3)$ are retained. This results in 64 candidate columns derived from x_3 and 49 candidate columns derived from x_{3*}. All these candidate columns are orthogonal to the first two design columns, because they are permutations of an orthogonal third column that preserve the elements of the first two columns. Therefore, to check whether the candidate columns are valid extensions of B, the extension algorithm only needs to test their orthogonality to the third column of B. In the example, six of the 64 candidate columns derived from x_3 and seven of the 49 candidate columns derived from x_{3*} are orthogonal to the third column of B. These candidate columns form the set $G(B,4)$. Appending them to B yields the set $E^+(B)$, which thus contains 13 4-column conference designs.

The set $G(B,4)$ includes every candidate for the fourth column that is smaller than the first three columns and orthogonal to them. Therefore, the corresponding set $E^+(B)$ is a complete set. In particular, it contains the LM0 representatives of all isomorphism classes and possibly additional designs that are not maximal with respect to the L0 ordering.

Algorithm 2: Pseudocode of the extension algorithm.

Input: $D, s(D,k-1), G(D,k)$: k-column conference design, row symmetry group for first $k-1$ columns, alternatives for column k

1 $s(D,k) \leftarrow$ `rowsymm`(D) `/* row symmetry group of design` D `preserving columns` $1,\ldots,k$ `*/`
2 $G(D,k+1) \leftarrow \varnothing$
3 $E^+(D) \leftarrow \varnothing$
4 **forall** $c \in G(D,k)$ **do**
5 Apply $s(D,k-1)$ to c and collect results in $G^-(D,k+1)$
6 $G^-(D,k+1) \leftarrow$ `maximal`$(G^-(D,k+1), s(D,k))$ `/* columns maximal with respect to` $s(D,k)$ `*/`
7 $G^-(D,k+1) \leftarrow$ `orthotest`$(G^-(D,k+1),$ `lastcol`$(D))$ `/* columns orthogonal to last column of` D `*/`
8 $G(D,k+1) \leftarrow G(D,k+1) \cup G^-(D,k+1)$
9 **forall** $u \in G(D,k+1)$ **do**
10 $F \leftarrow [D\ u]$ `/* append column` u `to design` D `*/`
11 $E^+(D) \leftarrow E^+(D) \cup \{F\}$

Output: Extended conference designs $E^+(D)$, row symmetry group $s(D,k)$, columns $G(D,k+1)$

Algorithm 3: Pseudocode of the reduction algorithm.

Input: Set of extended conference designs $C^+(n, k+1)$

$C^*(n, k+1) \leftarrow \varnothing$
forall $K \in C^+(n, k+1)$ **do**
  $r \leftarrow$ `LM0check`(K)
  if r is `True` **then**
    $C^*(n, k+1) \leftarrow C^*(n, k+1) \cup \{K\}$

Output: Minimum complete set of LM0 conference designs $C^*(n, k+1)$

3.3 Reduction algorithm

The reduction algorithm reduces a complete set of conference designs to a minimum complete set. It can be viewed as a modification of the check on lexicographic minimality for orthogonal arrays in Schoen et al. (2010). Pseudocode of the algorithm is shown in Algorithm 3. For each input design, Algorithm 3 calls the LM0 check function to test whether the design is in LM0 form. The function performs sign switches of rows and columns, column permutations and row permutations to find a larger design than the one being tested. As soon as it finds such a design, it returns a `False` and the design is discarded. If the function returns a `True`, the design is retained. Details of the LM0 check function are given in the Appendix.

To finalize the 16-row example, Algorithm 3 reduces the 13 designs in $C^+(16, 4)$ to the minimum complete set $C^*(16, 4)$ by removing nine designs that are not in LM0 form and retaining the four designs that are in LM0 form.

4 Results

In this section, we demonstrate the potential of the MCS algorithm by enumerating all conference designs with up to 24 rows, and thereby all DSDs with up to 49 runs. To the best of our knowledge, this is the largest catalog of DSDs to date. We report the numbers of isomorphism classes in Section 4.1. In Section 4.2, we show the computing times required by the open-source implementation and address some of its limitations. As an aid for practitioners, we identify the best DSDs in terms of the correlations among second-order effects in Section 4.3.

4.1 Numbers of isomorphism classes

We enumerated all LM0 conference designs with n rows and $4 \leq k \leq n$ columns, where n ranges from 4 to 24. Table 3 shows the numbers of conference designs in the minimum complete sets $C^*(n, k)$. These numbers, denoted by *tot*, equal the numbers of isomorphism classes of DSDs with $N = 2n + 1$ runs and k factors, which we denote by $D^*(N, k)$. For row sizes $n \leq 22$, Table 3 also shows the numbers of LM0 conference designs, *nd*, that cannot be extended to a conference matrix. The number of such designs increases substantially with the row size.

To a large extent, the results in Table 3 are new to the literature. There are only three exceptions. First, Schoen et al. (2019) proved that $n/4$ isomorphism classes exist for n-row 4-column conference designs when n is a multiple of 4, and that $(n-4)/2$ isomorphism classes exist for n-row 4-column conference designs when n is an odd multiple of 2. Our results for $k = 4$ agree with those of Schoen et al. (2019), as shown by the first row of Table 3. Second, our numbers of isomorphism classes for conference matrices match those given by Greig et al. (2006), and appear on the diagonal in Table 3. Finally, Núñez Ares and Goos (2020) enumerated orthogonal minimally aliased response surface designs with N rows, $k \leq 7$ columns, a specified number of zeros in each column and a specified number of zeros in each two-factor interaction column of the model matrix. Their designs have no repeat runs and no center points. Those with two zeros in each column and four zeros in each two-factor interaction column correspond with DSDs without a center point. Appendix C of their paper shows that they found two isomorphism classes for 5-factor DSDs in 20 unique runs. This number matches our number of conference designs with $n = 10$ rows and $k = 5$ columns.

For each row size n up to 20, Table 3 shows a contiguous set of column sizes k up to column size n for which $nd = 0$. This means that all designs for these column sizes can be extended to a conference matrix. Equivalently, the conference designs for these cases can be obtained by dropping columns from conference matrices of order n. By dropping different columns, different designs might be obtained. However, if a case includes just one isomorphism class, it is immaterial which columns are dropped from the conference matrix. In particular, for conferences designs with 6, 8, 10, 12, 14, 16 an 18 rows, the designs with at least 4, 5, 7, 9, 11, 15 and 15 columns, respectively, can be obtained by arbitrarily dropping columns from the corresponding conference matrices. This matches the results of Vazquez et al. (2020) on the best columns to drop

Table 3 Numbers of isomorphism classes of conference designs with $4 \leq n \leq 24$ rows and $4 \leq k \leq n$ columns. For $n \leq 22$, each entry has the form nd/tot; nd: number of isomorphism classes that cannot be derived from a conference matrix; tot: total number of isomorphism classes. For $n = 24$, only the total number tot is given. Superscripts refer to earlier results: ^{S}Schoen et al. (2019); ^{N}Núñez Ares and Goos (2020); ^{G}Greig et al. (2006).

k	n										
	4	6	8	10	12	14	16	18	20	22	24
4	$0/1^{S,G}$	$0/1^S$	$0/2^S$	$1/3^S$	$1/3^S$	$2/5^S$	$0/4^S$	$4/7^S$	$1/5^S$	$9/9^S$	6^S
5		0/1	0/1	$0/2^N$	0/2	2/5	0/7	9/13	0/15	28/28	30
6		$0/1^G$	0/1	0/2	1/5	7/12	7/30	82/92	55/219	637/637	1588
7			0/1	0/1	0/2	2/7	13/48	191/201	1171/1781	10962/10962	87929
8			$0/1^G$	0/1	0/2	2/7	21/77	234/251	4172/5292	70859/70859	1839474
9				0/1	0/1	0/3	3/42	30/47	2184/3640	78966/78966	8259167
10				$0/1^G$	0/1	0/3	1/37	9/26	698/2342	16865/16865	8667156
11					0/1	0/1	0/17	0/10	132/1589	101/101	4124471
12					$0/1^G$	0/1	0/13	0/10	42/1172	21/21	2397144
13						0/1	0/3	0/4	5/689	0/0	1806230
14						$0/1^G$	0/3	0/3	2/366	0/0	1353790
15							0/1	0/1	0/142	0/0	888475
16							$0/1^G$	0/1	0/57	0/0	499614
17								0/1	0/13	0/0	234006
18								$0/1^G$	0/5	0/0	91773
19									0/2	0/0	28730
20									$0/2^G$	0/0	7417
21										0/0	1377
22										$0/0^G$	232
23											19
24											9^G

from a DSD obtained by folding over a conference matrix.

For row sizes n up to 20, the number of conference designs that cannot be obtained from a conference matrix increases substantially with the row size. For 24-row conference designs, it was even computationally infeasible to check for each design whether it can be extended to a conference matrix. We conjecture that the number of designs for which this is not the case continues to increase with the row size.

Given that conference matrices of order 22 do not exist, a major contribution from our MCS algorithm is the enumeration of conference designs with 22 rows and up to 12 columns. The set $C^*(22, 12)$ even includes 21 isomorphism classes. For lack of order-22 conference matrices, the DSD construction of Xiao et al. (2012) does not allow the construction of 45-run DSDs. Our bottom-up enumeration remedies this problem: using the newly enumerated 22-row conference designs, it is now possible to build 45-run DSDs with up to 12 factors.

4.2 Computing times

In Table 4, we show the computing times of the minimum complete sets of conference designs with $4 \leq n \leq 24$ rows. The sets with $n \leq 22$ were obtained using a laptop with Windows 10 as operating system, and an Intel(R) Core(TM) i7-8550U central processor unit. We used Python 3.7.4 in conjunction with the oapackage version 2.6.7 (Eendebak and Vazquez, 2019).

Table 4 shows that the sets with $n \leq 20$ rows can be enumerated in a matter of seconds, while the 22-row sets can be enumerated in a matter of minutes. In addition, it took 65 seconds to enumerate the 24-row conference designs with up to 7 columns. After extending a few 7-column designs, we predicted that the total number of 8-column designs would exceed one million. This made it clear to us that a computer cluster would be required to complete the enumeration of 24-row conference designs. Table 4 reports the approximate computing time for the enumeration of the $n = 24$ sets on the computer cluster of the University of Antwerp, because the enumeration had to be split over multiple small computer jobs.

Table 4 Computing times (in seconds) for all conference designs with $4 \leq n \leq 24$ rows and $4 \leq k \leq n$ columns.

n	4	6	8	10	12	14	16	18	20	22	24
time	< 0.01	< 0.01	< 0.01	< 0.01	0.03	0.06	0.4	1	35	223	(6 days)

Tests involving conference designs with $n > 24$ rows all point to an exponential increase in the number of designs. These tests and the approximate computing time for the 24-row series suggest that series with $n > 24$ can only be enumerated if we impose extra restrictions such as an upper bound on the correlation between two-factor interaction contrast vectors. A similar approach allowed Schoen et al. (2017) to enumerate strength-2 orthogonal arrays with 32 runs.

4.3 Definitive screening designs that minimize the correlation among contrast vectors for second-order effects

In their studies of DSDs, Jones and Nachtsheim (2011) and Vazquez et al. (2020) point out that the contrast vectors for linear effects (LEs) are uncorrelated with those of quadratic effects (QEs) and two-factor interactions (TFIs). Therefore, first-order effects are not aliased with second-order effects. However, contrast vectors of QEs and TFIs can be correlated. A large absolute correlation implies that two second-order effects are aliased to a large extent, which affects the potential of any DSD to fit a model that includes some QEs and TFIs. In this section, we present DSDs for 5–12 factors, obtained from the enumerated conference designs, that minimize the correlation among contrast vectors of second-order effects in some sense. In particular, we identify the best designs according to the G-aberration criterion (Schoen et al., 2019) and the β-aberration criterion (Cheng and Ye, 2004).

Vazquez et al. (2020) showed that, for a given run size N, the only features of the information matrix of a DSD that are not completely determined by the run size are its off-diagonal elements corresponding to inner products of contrast vectors of two TFIs involving four different factors. The absolute values of these inner products are called J_4 characteristics. For any subset of four factors, there is one J_4 characteristic. Schoen et al. (2019) showed that it is of the form $2n - 8q$, $q = 1, \ldots, \lfloor n/4 \rfloor$, where $\lfloor x \rfloor$ is the integer part of x and n is the row size of the corresponding conference design. When considering a four-factor DSD to perform an experiment, it is best to pick one that minimizes the single J_4 characteristic. This is because the absolute correlation between contrast vectors of TFIs that involve all four factors equals $J_4/(2n-4)$. Small correlations between such contrast vectors result in smaller correlations for the ordinary least squares estimators in any model involving more than one TFI and a smaller bias in any model involving only one of the two-factor interactions.

For cases with more than four factors, Schoen et al. (2019) proposed to count the J_4 characteristics of $2n - 8q$, $q = 1, \ldots, \lfloor n/4 \rfloor$, summarize the counts in a frequency vector F_4 and order alternative DSDs based on that vector. The first frequency in the F_4 vector corresponds to $q = 1$, and therefore to the largest possible J_4 characteristic (and thus to the most severe absolute correlation between contrast vectors of TFIs). The last of its frequencies corresponds to $q = \lfloor n/4 \rfloor$, and therefore to the smallest possible J_4 characteristic (and thus to the least severe correlation). Schoen et al. (2019) proposed to rank DSDs in ascending order of the F_4 vector's first entry. Designs with the same first entry are sorted in ascending order of the second entry. This process continues until a unique order has been established or all entries of the F_4 vector have been considered. The G-aberration of a DSD is its rank after the sorting procedure. A design with rank 1 has a minimum G-aberration. Note that there may be more than one minimum G-aberration design.

The G-aberration criterion for DSDs prioritizes the minimization of the worst absolute correlations between contrast vectors of TFIs involving four factors. As an alternative, one could minimize a measure for the total correlation among all contrast vectors involving second-order effects. A suitable criterion for this purpose is the first nonzero entry of the β word length pattern (Cheng and Ye, 2004), which we denote by $\beta_{4,\text{tot}}$. Further terms in the β word length pattern quantify correlations involving higher-order effects. Since DSDs are used in practice to study first- and second-order effects, we focus on a correlation-based criterion related to these kinds of effects and ignore entries in the β word length pattern involving higher-order effects. However, the completeness of our catalogs permits any researcher to find the best DSDs according to criteria quantifying these higher-order effects.

To calculate $\beta_{4,\text{tot}}$ for a k-factor DSD, we normalize the k LE and k QE contrast vectors to have a mean of zero and a length of $\sqrt{N}$. The scalar $\beta_{4,\text{tot}}$ is the sum of three components. The first component, which we

denote by $\beta_{4,lll}$, is calculated by taking the element-wise products of all $\binom{k}{4}$ sets of four normalized LE contrast vectors. The elements of the resulting vectors are summed, the sums are squared and the squares are added and divided by N^2. As the element-wise product of two LE contrast vectors is a contrast vector of a TFI, the $\beta_{4,lll}$ value is proportional to the sum of squared correlations among TFI contrast vectors that involve four different factors.

The second component of $\beta_{4,\text{tot}}$, which we denote by $\beta_{4,qq}$, is calculated by taking the inner products of all pairs of normalized QE contrast vectors, squaring each inner product, adding the results and dividing by N^2. The $\beta_{4,qq}$ value is thus the sum of squared correlations among QE contrast vectors.

The third component, which we denote by $\beta_{4,llq}$, is calculated by taking the element-wise products of one normalized QE contrast vector and two normalized LE contrast vectors. The elements of the resulting vectors are summed, and the squared sums are added and divided by N^2. The $\beta_{4,llq}$ value is thus proportional to the sum of squared correlations between a QE contrast vector and a TFI contrast vector.

Table 5 shows key characteristics of minimum G-aberration and minimum β-aberration DSDs with $5 \leq k \leq 12$ factors and $N \leq 49$ runs. The table therefore covers most of the run sizes and numbers of factors that are likely to be used in practice. For each design, the table shows the maximum correlation among TFI contrast vectors involving four different factors, $\rho_{\max}$, along with the frequency f of the corresponding J_4 characteristics. It can be shown that these J_4 characteristics contribute an amount of $\beta_{4,\rho_{\max}} = f\rho_{\max}^2(N-5)^2N^2/(N-3)^4$ to $\beta_{4,\text{tot}}$. The table shows $\beta_{4,\rho_{\max}}$ along with $\beta_{4,lll}$ and $\beta_{4,\text{tot}}$.

To illustrate how the table can be used to assess DSDs, we use the minimum G-aberration DSD with $N = 2n + 1 = 17$ runs and five factors as an example. This DSD has a maximum correlation among TFI contrast vectors of 0.667. The corresponding J_4 characteristic, which we denote by $J_4^{\max}$, is calculated as $J_4^{\max} = (2n-4)\rho_{\max} = 8$. The five-factor design has two such J_4 characteristics, which give rise to a total of six pairs of correlated TFI contrast vectors that involve four different factors. Three J_4 characteristics equal 0 and give rise to a total of nine pairs of uncorrelated TFI contrast vectors involving four different factors. Therefore, $\beta_{4,lll} = \beta_{4,\rho_{\max}}$. In total, the contribution of TFI contrast vectors that involve four different factors to $\beta_{4,\text{tot}}$ equals 0.96.

For the 17-run five-factor DSD in Table 5, there are a total of 30 pairs of TFI contrast vectors with a common factor. The correlation within such pairs equals $\pm 2/(2n-4) = \pm 0.167$ (Vazquez et al., 2020). This correlation is not accounted for in the G-aberration criterion, because it is the same for all designs with the same run size. However, it does contribute to the component $\beta_{4,llq}$ of $\beta_{4,\text{tot}}$, because the correlation among two TFI contrast vectors with a common factor is proportional to the correlation between the QE contrast vector corresponding to the common factor and the TFI contrast vector involving the remaining two factors and thus to the component $\beta_{4,llq}$ of $\beta_{4,\text{tot}}$. The $\beta_{4,\text{tot}}$ value of 5.54 for the 17-run 5-factor DSD indicates that the aliasing among TFIs involving four different factors represent only a small part of the overall aliasing among second-order effects. In general, the values of $\beta_{4,lll}/\beta_{4,\text{tot}}$ in Table 5 range from 0 to 0.5.

The table shows 15 cases for which the minimum G-aberration and β_4-aberration designs differ. The minimum G-aberration DSDs for these cases have '-1' appended to the number of factors k, while the DSDs with the smallest $\beta_{4,\text{tot}}$ value have '-2' appended to k. However, the differences in the $\beta_{4,\text{tot}}$ value are minor for these cases, while the difference in maximum correlations or the frequencies of the corresponding J_4 characteristics are more substantial. For this reason, we prefer the minimum G-aberration DSDs to the minimum β-aberration DSDs.

5 Discussion

In this paper, we presented methodology to obtain minimum complete sets of conference designs with given numbers of rows n and columns k, and thereby DSDs with k factors and a run size $N = 2n+1$. The innovative featurs of our methodology are (1) the enumeration of conference designs with $k+1$ columns from a minimum complete set of conference designs with k columns, (2) a clearly defined representative of an isomorphism class and (3) a fast rejection test to discard conference designs that differ from that representative.

Our methodology is implemented in an open source suite of Python procedures. We demonstrated its effectiveness by producing all non-isomorphic conference designs with up to 24 rows. Only 0.06% of these designs have row sizes smaller than or equal to 20, 0.59% of them have row size $n = 22$, and the remaining 99.35% have row size 24. Based on these findings, we conjecture that there are many more 26-row conference designs than 24-row conference designs. In addition, tests with conference designs with more than 24 rows suggest an exponential increase in the number of isomorphism classes and the running time. Therefore, a complete enumeration of conference designs with more than 24 rows will be infeasible using our algorithms.

Table 5 Key characteristics of minimum G-aberration and minimum β-aberration DSDs with $5 \leq k \leq 12$ factors and $N \leq 49$ runs. $\rho_{\max}$): maximum correlation among two-factor interaction contrast vectors; f: number of corresponding J_4 characteristics; $\beta_{4,\rho_{\max}}$: contribution of maximum correlations among two-factor interaction contrast vectors involving 4 different factors to $\beta_{4,\text{tot}}$; $\beta_{4,llll}$: contribution of all correlations among two-factor interaction contrast vectors involving 4 different factors to $\beta_{4,\text{tot}}$; $\beta_{4,\text{tot}}$: first non-zero entry of β word length pattern.

N	k	$\rho_{\max}$	f	$\beta_{4,\rho_{\max}}$	$\beta_{4,llll}$	$\beta_{4,\text{tot}}$	N	k	$\rho_{\max}$	f	$\beta_{4,\rho_{\max}}$	$\beta_{4,llll}$	$\beta_{4,\text{tot}}$
17	5	0.667	2	0.96	0.96	5.54	37	5	0.125	5	0.08	0.08	2.23
	6	0.667	6	2.89	2.89	11.86		6	0.125	15	0.25	0.25	4.16
	7	0.667	14	6.74	6.74	22.25		7	0.375	9	1.33	2.02	8.47
	8	0.667	28	13.48	13.48	38.09		8	0.375	23	3.39	4.43	14.34
21	5	0.25	5	0.34	0.34	3.85		9	0.375	45	6.64	9.41	23.82
	6	0.75	2	1.21	2.08	8.87		10-1	0.375	79	11.65	13.8	33.91
	7	0.75	5	3.02	5.04	16.66		10-2	0.625	4	1.64	13.67	33.78
	8	0.75	10	6.05	10.08	28.4		11	0.375	128	18.88	22.19	49.33
	9	0.75	18	10.89	18.15	45.33		12	0.375	192	28.32	33.29	68.92
	10	0.75	30	18.15	30.25	68.77	41	5	0.222	3	0.15	0.15	2.17
25	5	0.4	3	0.51	0.51	3.45		6	0.222	9	0.46	0.46	4.1
	6	0.4	9	1.54	1.54	7.11		7	0.222	25	1.29	1.5	7.44
	7	0.4	23	3.93	3.93	13.38		8	0.444	4	0.83	3.77	12.83
	8	0.4	46	7.85	7.85	22.65		9	0.444	12	2.48	7.02	20.15
	9	0.4	84	14.34	14.34	36.18		10	0.444	20	4.13	12.18	30.43
	10	0.4	140	23.91	23.91	54.72		11-1	0.444	36	7.43	19.61	44.16
	11	0.4	220	37.57	37.57	79.54		11-2	0.667	3	1.39	19.4	43.95
	12	0.4	330	56.35	56.35	111.88		12-1	0.444	56	11.56	29.62	61.78
29	5	0.167	5	0.15	0.15	2.72		12-2	0.667	6	2.79	29.25	61.41
	6	0.5	3	0.8	1.15	5.96	45	5	0.1	5	0.05	0.05	1.96
	7	0.5	7	1.86	2.68	10.76		6	0.1	15	0.16	0.16	3.57
	8	0.5	18	4.77	6.3	18.86		7	0.3	8	0.75	1.03	6.57
	9	0.5	33	8.75	11.48	29.93		8-1	0.3	25	2.34	2.81	11.22
	10	0.5	57	15.11	19.61	45.54		8-2	0.5	2	0.52	2.56	10.97
	11	0.5	90	23.85	30.92	66.11		9-1	0.5	2	0.52	6.48	18.6
	12	0.5	135	35.78	46.38	92.82		9-2	0.5	6	1.56	5.81	17.93
33	5	0	0	0	0	2.32		10-1	0.5	6	1.56	11.68	28.48
	6	0.286	6	0.52	0.52	4.81		10-2	0.5	30	7.81	9.68	26.48
	7-1	0.571	4	1.38	2.41	9.55		11-1	0.5	24	6.25	17.26	39.79
	7-2	0.857	2	1.55	2.07	9.21		11-2	0.7	3	1.53	17.01	39.54
	8-1	0.571	10	3.44	5.33	16.36		12-1	0.5	45	11.71	27.39	56.84
	8-2	0.857	1	0.77	4.82	15.84		12-2	0.7	3	1.53	25.9	55.34
	9-1	0.857	1	0.77	10.33	26.44	49	5	0.182	2	0.07	0.07	1.9
	9-2	0.857	6	4.65	9.29	25.4		6	0.182	6	0.21	0.21	3.44
	10-1	0.857	3	2.32	16.69	39.25		7	0.364	3	0.41	0.82	6.04
	10-2	0.857	6	4.65	16	38.56		8	0.364	11	1.51	2.27	10.14
	11-1	0.857	7	5.42	25.99	56.52		9	0.364	27	3.71	5.01	16.32
	11-2	0.857	10	7.74	25.81	56.34		10-1	0.364	49	6.73	8.79	24.41
	12	0.857	12	9.29	39.24	79.42		10-2	0.545	3	0.93	8.72	24.34
								11-1	0.364	84	11.53	14.83	35.73
								11-2	0.545	1	0.31	14.69	35.59
								12	0.364	144	19.77	22.24	49.49

One way to construct conference designs with row sizes larger than 24 could be to adapt the enumeration approach of Schoen et al. (2017) for 32- and 36-run orthogonal arrays with maximum J_3 characteristic values

of 4 to enumerate large conference designs with certain maximum J_4 characteristics. However, as practical experiments rarely include more than 24 factors and more than 49 runs, we believe that our present catalog of DSDs includes all designs likely to be needed by practitioners.

With minor modifications, our methodology can be used to enumerate weighing designs (Georgiou et al., 2014) as well. Denoting an $n \times k$ weighing design with $\mathbf{W}$, the defining property of such a design is that $\mathbf{W}^T\mathbf{W} = w\mathbf{I}_k$, where $w \leq n-1$ denotes the number of ± 1 elements in each column of the design. Therefore, conference designs are a special kind of weighing designs. By folding over weighing designs, we obtain designs which contain more zeros for every factor than DSDs in case $w < n-1$. In that case, they provide more precise estimates for the quadratic effects than the DSDs. Therefore, an interesting subject for further research is the enumeration of weighing designs and the characterization of folded over weighing designs.

The classification criteria we used are G-aberration and β-aberration. Given a run size N and a number of factors k, designs that minimize either of these aberration criteria should perform best in terms of statistical modeling. For the case of two-level orthogonal arrays, Cheng et al. (2002) established a link between the B_3 and B_4 counts of a design and the D-efficiency of a model including all main effects and some two-factor interactions. For fold-over orthogonal designs, whose B_3 counts equal 0, this D-efficiency is inversely related to the B_4 count, which in two-level orthogonal arrays equals $\beta_{4,\text{tot}}$. We expect this result for two-level designs to hold also for DSDs. However, as DSDs also can estimate quadratic effects, model building is more complex than for two-level screening designs. For this reason, we plan to evaluate all enumerated DSDs in terms of efficiency-based criteria in future research. Of particular interest will be the evaluation of D-efficiencies and average prediction variances for full second-order models in projections of the DSDs into fewer factors. Indeed, as the enumeration is complete, one can evaluate the DSD series with any criterion that comes to mind.

Finally, one might consider using optimization algorithms, such as coordinate- and point-exchange algorithms to construct optimal designs, as a computationally less intensive method to generate DSDs. However, such algorithms cannot guarantee reaching the best solution for even a single criterion and fails to find orthogonal designs in many cases (Cuervo et al., 2016). In contrast, our complete enumeration of definitive screening designs with up to 49 runs gives definitive answers about which DSD is best with respect to any criterion for practically relevant run sizes.

Acknowledgements

The authors are grateful to two anonymous referees, whose comments were helpful to clarify the text.

Conflict of interest

The authors declare that they have no conflict of interest.

Supplemental information

- Sample code.py: python code to enumerate and output conference designs.
- Table 5 designs.zip: conference designs to build all DSDs evaluated in Table 5.

Appendix: LM0 check function

Algorithm 4: Pseudocode of the LM0 check.

Input: Design K

for $i = 1, \ldots, k+1$ **do**
  $A \leftarrow$ design K with columns 1 and i interchanged `/* Select first column */`
  Sort the rows of design A such that $a_{1,1} = 0$
  Apply sign switches in row $2, \ldots, n$ such that $a_{i,1} = 1$ for $i > 1$ `/* Make first column lexicographically maximal */`
  foreach $s_r \in \{1, -1\}$ **do**
    $T_1 \leftarrow$ design A with row 1 multiplied by s_r `/* Sign switches in first row */`
    Apply sign switches in column $2, \ldots, k+1$ such that $T_{1,j} = 1$ for $j > 1$
    $r \leftarrow$ `continuation`$(T_1, K, 2)$ `/* Call the continuation function in Algorithm 5 */`
    if r is `False` **then**
      Return `False`
Return `True`

Output: `True` if the design is in LM0 form, `False` otherwise

The reduction algorithm in Section 3.3 calls the LM0 check function given in Algorithm 4, which itself calls a continuation function given in Algorithm 5.

The purpose of the LM0 check function is to determine whether or not an input design is of LM0 form. To this end, it performs column permutations, row permutations, sign switches of rows, and sign switches of columns to check whether these operations result in a lexicographically larger design. As soon as a lexicographically larger design is found, the procedure stops and returns a `False` for the input design. The sign switches and a small portion of the permutations are addressed in Algorithm 4. All remaining column permutations are addressed in Algorithm 5.

In lines 1 and 2, Algorithm 4 considers all designs A in which column 1 is interchanged with any of the columns $1, \ldots, (k+1)$, thereby including the option that the first column stays in its original position. Next, in lines 3 and 4, the rows are sorted and sign switches of the rows are applied until the first column is in the largest form possible: a zero entry in the first position $a_{1,1}$ and entries of 1 in all other positions $a_{i,1}$, where $i > 1$. At this point, the design can only be made lexicographically larger by applying a sign switch to row 1 or to columns $2, \ldots, (k+1)$ so that all columns other than the first start with a 1. These options are explored in lines 5–8 of Algorithm 4. At this point, all sign switches that might make the current design lexicographically larger are exhausted. However, there may still be column permutations that result in a lexicographically larger design. That part of the LM0 check is performed by Algorithm 5.

Algorithm 5 carries out additional column permutations as well as the actual LM0 check between the original input design K and the design modified by the sign switches and permutations of Algorithms 4 and 5. The input of Algorithm 5 is the original input design K, the modified design and the leftmost column c that is to be permuted with columns $c, c+1, \ldots, k+1$. In its line 2, Algorithm 5 starts by conducting column permutations involving the current column c and sorting the rows such that earlier columns remain unchanged. This qualification is necessary because the 'larger than' operation works column by column. If the sorting would change the row order of the first $c-1$ columns, the current design would not be as large as possible.

After the sorting, the current column is tested against the corresponding column in the original design K (lines 4–9). If the current column is lexicographically larger, then the original design is not of LM0 form. If both columns are equal, further column permutations are invoked by calling the algorithm recursively using a new current column. Finally, if the current column it is smaller than the corresponding column in K, a new iteration of the **for** loop is started.

Algorithm 5: Pseudocode of the continuation function.

Input: Design $T_{(c-1)}$, original design K, current column c

1 **for** $c_x = c, \ldots, k+1$ **do**
2 $T_{c_x} \leftarrow$ design $T_{(c-1)}$ with column c and c_x interchanged
3 Sort the rows of T_{c_x} according to LM0 ordering of the new column c and leaving columns $1, \ldots, (c-1)$ unchanged
   `/* Compare column` c `of` T_{c_x} `with column` c `of` K `using LM0 ordering */`
4 **if** $T_{c_x}(c) \succ K(c)$ **then**
5 Return `False`
6 **if** $T_{c_x}(c) = K(c)$ **then**
   `/* Recurse one level deeper */`
7 $r \leftarrow$ `continuation`$(T_{c_x}, K, c+1)$
8 **if** r is `False` **then**
9 Return `False`
10 Return `True`

Output: `True` if $\forall c, c \le x \le k+1 : T_{c_x} \prec K$, `False` otherwise

References

Atkinson, A., Donev, A., and Tobias, R. (2007). *Optimum Experimental Designs, With SAS*. OUP Oxford.

Cheng, C., Deng, L., and Tang, B. (2002). Generalized minimum aberration and design efficiency for nonregular fractional factorial designs. *Statistica Sinica*, 12:991–1000.

Cheng, S. W. and Ye, K. Q. (2004). Geometric isomorphism and minimum aberration for factorial designs with quantitative factors. *The Annals of Statistics*, 32:2168–2185.

Colbourn, C. J. and Dinitz, J. H. (2006). *Handbook of Combinatorial Designs*. 2nd edition; Boca Raton: Chapman and Hall/CRC.

Cuervo, D. P., Goos, P., and Sörensen, K. (2016). Optimal design of large-scale screening experiments: a critical look at the coordinate-exchange algorithm. *Statistics and Computing*, 26:15–28.

Dougherty, S., Simpson, J. R., Hill, R. R., Pignatiello, J. J., and White, E. D. (2015). Effect of heredity and sparsity on second-order screening design performance. *Quality and Reliability Engineering International*, 31:355–368.

Eendebak, P. and Vazquez, A. (2019). OApackage: A Python package for generation and analysis of or-

thogonal arrays, optimal designs and conference designs. *Journal of Open Source Software*, 4:1097.

Fidaleo, M., Lavecchia, R., Petrucci, E., and Zuorro, A. (2016). Application of a novel definitive screening design to decolorization of an azo dye on boron-doped diamond electrodes. *International Journal of Environmental Science and Technology*, 13:835–842.

Georgiou, S. D., Stylianou, S., and Aggarwal, M. (2014). Efficient three-level screening designs using weighing matrices. *Statistics*, 48:815–833.

Greig, M., Haanpää, H., and Kaski, P. (2006). On the coexistence of conference matrices and near resolvable $2-(2k+1,k,k-1)$ designs. *Journal of Combinatorial Theory Series A*, 113:703–711.

Jones, B. and Lanzerath, M. (2021). A novel application of a definitive screening design: A case study. *Quality Engineering*, 33:563–569.

Jones, B. and Nachtsheim, C. J. (2011). A class of three-level designs for definitive screening in the presence of second-order effects. *Journal of Quality Technology*, 43:1–15.

Jones, B. and Nachtsheim, C. J. (2013). Definitive screening designs with added two-level categorical factors. *Journal of Quality Technology*, 45:121–129.

Jones, B. and Nachtsheim, C. J. (2017). Effective design-based model selection for definitive screening designs. *Technometrics*, 59:319–329.

Libbrecht, W., Deruyck, F., Poelman, H., Verberckmoes, A., Thybaut, J., De Clercq, J., and Van Der Voort, P. (2015). Optimization of soft templated mesoporous carbon synthesis using Definitive Screening Design. *Chemical Engineering Journal*, 259:126–134.

Maestroni, B. M., Vazquez, A. R., Avossa, V., Goos, P., Cesio, V., Heinzen, H., Riener, J., and Cannavan, A. (2018). Ruggedness testing of an analytical method for pesticide residues in potato. *Accreditation and Quality Assurance*, 23:303–316.

Mee, R. W., Schoen, E. D., and Edwards, D. E. (2017). Selecting an orthogonal or non-orthogonal two-level design for screening. *Technometrics*, 59:305–318.

Nachtsheim, A. C., Shen, W., and Lin, D. K. J. (2017). Two-level augmented definitve screening designs. *Journal of Quality Technology*, 49:93–107.

Nguyen, N.-K. and Pham, T.-D. (2016). Small mixed-level screening designs with orthogonal quadratic effects. *Journal of Quality Technology*, 48:405–414.

Nguyen, N.-K. and Stylianou, S. (2013). Constructing definitive screening designs using cyclic generators. *Journal of Statistical Theory and Practice*, 7:713–724.

Núñez Ares, J. and Goos, P. (2020). Enumeration and multicriteria selection of orthogonal minimally aliased response surface designs. *Technometrics*, 62:21–36.

Olsen, R. E., Bartholomew, C. H., Enfield, D. B., Lawson, J. S., Rohbock, N., Scott, B. S., and Woodfield, B. F. (2014). Optimizing the synthesis and properties of Al-modified anatase catalyst supports by statistical experimental design. *Journal of Porous Materials*, 21:827–837.

Phoa, F. K. H. and Lin, D. K. J. (2015). A systematic approach for the construction of definitive screening designs. *Statistica Sinica*, 25:853–861.

Renzi, P., Kronig, C., Carlone, A., Eröksüz, S., Berkessel, A., and Bella, M. (2014). Kinetic resolution of oxazinones: Rational exploration of chemical space through the design of experiments. *Chemistry - A European Journal*, 20:11768–11775.

Schoen, E. D., Eendebak, P. T., and Goos, P. (2019). A classification criterion for definitive screening designs. *The Annals of Statistics*, 47:1179–1202.

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

Schoen, E. D., Vo-Thanh, N., and Goos, P. (2017). Two-level orthogonal screening designs with 24, 28, 32, and 36 runs. *Journal of the American Statistical Association*, 112:1354–1369.

Tai, M., Ly, A., Leung, I., and Nayar, G. (2015). Efficient high-throughput biological process characterization: Definitive screening design with the Ambr250 bioreactor system. *Biotechnology Progress*, 31:1388–1395.

Vazquez, A. R., Goos, P., and Schoen, E. D. (2020). Projections of definitive screening designs by dropping columns: Selection and evaluation. *Technometrics*, 62:37–47.

Xiao, L., Lin, D. K., and Bai, F. (2012). Constructing definitive screening designs using conference matrices. *Journal of Quality Technology*, 44:1–7.