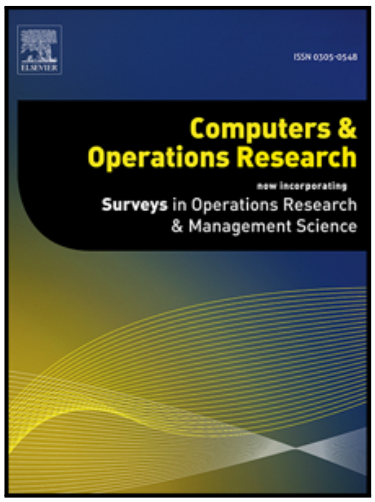
Symmetry breaking in mixed integer linear programming formulations for blocking two-level orthogonal experimental designs



Symmetry Breaking in Mixed Integer Linear Programming Formulations for Blocking Two-level Orthogonal Experimental Designs

Nha Vo-Thanh, Raf Jans, Eric D. Schoen, Peter Goos

PII: S0305-0548(18)30084-4
DOI: 10.1016/j.cor.2018.04.001
Reference: CAOR 4442

To appear in: *Computers and Operations Research*

Received date: 15 September 2016
Revised date: 31 January 2018
Accepted date: 3 April 2018

Please cite this article as: Nha Vo-Thanh, Raf Jans, Eric D. Schoen, Peter Goos, Symmetry Breaking in Mixed Integer Linear Programming Formulations for Blocking Two-level Orthogonal Experimental Designs, *Computers and Operations Research* (2018), doi: 10.1016/j.cor.2018.04.001

Highlights

- Orthogonal blocking is a challenging problem in design of experiments.
- The problem can be tackled using integer linear programming (ILP).
- Symmetry breaking formulations speed up the solution of the ILP.
- For some instances, an asymmetric representatives formulation is extremely fast.
- Our work remains useful even though solvers implement symmetry breaking too.

Symmetry Breaking in Mixed Integer Linear Programming Formulations for Blocking Two-level Orthogonal Experimental Designs

Nha Vo-Thanh
Institute of Crop Science, University of Hohenheim, Germany
Faculty of Applied Economics, University of Antwerp, Belgium

Raf Jans
Department of Logistics and Operations Management, HEC Montréal and GERAD, Canada

Eric D. Schoen
Faculty of Applied Economics, University of Antwerp, Belgium
TNO, Zeist, Netherlands

Peter Goos
Faculty of Bioscience Engineering and Leuven Statistics Research Centre (LSTAT), KU Leuven, Belgium
Faculty of Applied Economics and StatUa Center for Statistics, University of Antwerp, Belgium

April 5, 2018

Abstract

Two-level orthogonal designs play an important role in industrial screening experiments, in which the primary goal is to identify the treatment factors with the largest main effects on the output of a process or the performance of a product. Often, the experimental tests suggested by the orthogonal designs cannot be performed on a single day or using a single batch of raw material. This causes day-to-day or batch-to-batch variation in the experimental results, and necessitates the use of orthogonal blocking patterns. These blocking patterns ensure that the factors' main effects can be estimated independently from the day-to-day or batch-to-batch variation. Finding an optimal orthogonal blocking pattern for an orthogonal design is a major challenge. Recently, mixed integer linear programming has been used for that purpose. While this approach is promising, computational experiments have indicated it is quite slow. We show how to speed up the mixed integer linear programming approach by adding symmetry breaking constraints to the formulation, and study the usefulness of an asymmetric representatives formulation. In other words, we introduce state-of-the-art symmetry breaking approaches in optimal experimental design. We perform extensive computational experiments to test which combinations of symmetry breaking constraints work best. Throughout, we consider two kinds of symmetry: symmetry due to the fact that the blocks can be relabeled without affecting the quality of the blocking pattern, and symmetry due to replicated test combinations.

KEY WORDS: Asymmetric Representatives Formulation; Confounding; Symmetry Breaking Constraints; Integer Linear Programing; Orthogonal Blocking

1 Introduction

1.1 Experimental designs

Experimental designs play an important role in industrial engineering, as they are key tools in the context of product and process innovation or improvement. More specifically, experimental designs define a plan for a systematic study of the relationship between one or more outputs of a process and the inputs to that process, or between one or more measures of a product's performance and the characteristics of that product. Generally, we use the name factor or treatment factor for each process input or for each product characteristic studied in an experiment. In the initial phase of a study, experiments typically involve a large number of factors and two levels are used for each factor.

For example, a car tire manufacturer may perform an experiment to investigate how the wear of tires depends on twelve factors, and use two levels for each factor: Rubber Compound (A/B), Number of Ribs (small/large), Shoulder Block Position (low/high), Center Line Cut Depth (low/high), Center Line Cut Width (low/high), Shoulder Cut Width (low/high), Center Line Cut Angle (small/large), Shoulder Cut Angle (low/high), Center Line Cut Through (yes/no), Shoulder Cut Through (yes/no), Center Line Additional Sipe (yes/no), and Shoulder Additional Sipe (yes/no). A full factorial experiment using these twelve two-level factors would require $2^{12} = 4096$ runs or tests, which is practically infeasible due to the high costs and time required.

Commonly used alternatives to full factorial designs for the treatment factors, which require substantially fewer experimental tests, are fractional factorial designs and Plackett-Burman designs (see, e.g., Montgomery, 2009; Wu and Hamada, 2009), as well as other two-level orthogonal arrays (see, e.g., Hedayat et al., 1999; Mee, 2009; Schoen et al., 2017). A two-level orthogonal array with n factors, N runs and strength t is an $N \times n$ matrix of -1s and 1s. Each column corresponds to a factor and involves as many -1s as 1s. Moreover, in every set of t columns, each t-tuple of -1s and 1s occurs equally often. We denote the set of non-isomorphic two-level orthogonal arrays of strength t by $\mathrm{OA}(N; 2^n; t)$. The reason why orthogonal arrays are popular experimental designs is that they allow for independent estimates of the factors' main effects on the process output or the product performance. Orthogonal arrays of strength 3 are more attractive than orthogonal arrays of strength 2 because they ensure that the main effect estimates are not biased whenever there are two-factor interaction effects. Orthogonal arrays of strength 4 do not only allow the independent estimation of the main effects, but also of the two-factor interaction effects.

1.2 Need for blocking

A drawback of orthogonal arrays with a strength larger than 2 is that they involve a larger number of runs. Experiments based on orthogonal arrays with a larger number of runs typically span multiple days, or require multiple batches of raw material, multiple operators or machines. It is common to observe day-to-day, batch-to-batch, operator-to-operator or machine-to-machine variation in experimental results. For this reason, experimental design textbooks emphasize the importance of using orthogonal blocking arrangements. An orthogonal blocking arrangement is essentially an assignment of the experimental tests to the different days, batches, operators or machines used during the experiment, so that the estimates of the factors' main effects are not affected by the day-to-day, batch-to-batch, operator-to-operator or machine-to-machine variation. Ideally, the estimates of the interaction effects are not affected by these sources of variation either. For experiments with many factors, it is, however, often impossible to find blocking arrangements with this desirable property for the interaction effects' estimates. Therefore, experimenters often have to content themselves with blocking arrangements that are orthogonal for the main effects, but not for the interaction effects.

Generally, the day, batch, operator or machine is referred to as a blocking factor. The blocking factor assigns the experimental tests defined by the levels of the treatment factors to several blocks. In this article, we denote the number of blocks by b. One way to obtain orthogonal blocking arrangements involving b blocks of N/b experimental tests is to construct an orthogonal array of the type $\mathrm{OA}(N; b \times 2^n; 2)$, involving one b-level factor and n two-level factors. Such an orthogonal array ensures that the main effects' estimates are not affected by the block-to-block variation. The challenge then is to enumerate all possible orthogonal arrays of the type $\mathrm{OA}(N; b \times 2^n; 2)$ (for instance, using the algorithm of Schoen et al. (2010)), and identify the ones for which the interaction effects' estimates are influenced as little as possible by the block-to-block

variation. This kind of procedure has been used successfully by Schoen et al. (2013) and Schoen et al. (2018). Generally, the two-level part of an array of the type OA$(N; b \times 2^n; 2)$ is referred to as the treatment design, because it defines the combinations of levels of the treatment factors to be tested in the N runs of the experiment.

Unfortunately, for run sizes $N \geq 32$, it is computationally infeasible to generate and explore complete catalogs of non-isomorphic orthogonal arrays of the type OA$(N; b \times 2^n; 2)$. Therefore, searching blocking arrangements of arrays of the type OA$(N; 2^n; 2)$ using the methods of Schoen et al. (2018) is not feasible for these run sizes. Instead, one could explore catalogs of arrays of the type OA$(N; b \times 2^n; 3)$. However, the usefulness of this approach is limited because it requires both the full orthogonal array (including the b-level blocking factor) and the two-level part of the array (the treatment design) to be of strength 3. This requirement is too strict because experimenters are generally not interested in interactions between the blocking factor and the treatment factors. Therefore, it is sufficient for the full orthogonal array to have strength 2, while the two-level treatment design has strength 3. At present, no methodology is available for enumerating all possible ways in which all non-isomorphic strength-3 treatment designs of the type OA$(N; 2^n; 3)$ can be embedded in an orthogonal blocking arrangement of the type OA$(N; b \times 2^n; 2)$. Consequently, it is unknown how to optimally arrange, for example, the runs of the large number of attractive two-level treatment designs of strength 3 identified by Schoen and Mee (2012) in blocks.

To circumvent the above enumeration problems, Sartono et al. (2015a) presented a mixed integer linear programming formulation for arranging any given treatment design of the type OA$(N; 2^n; t)$ in b blocks such that the complete design is an orthogonal array of the type OA$(N; b \times 2^n; 2)$. This procedure thus guarantees that the estimators of the main effects are independent of the blocking factor. In addition, the procedure minimizes the dependence of the two-factor interaction effects' estimates on the blocking factor.

1.3 The problem of symmetry

The main weakness of the mixed integer linear programming approach is the long computing time required to obtain optimal blocking arrangements. This is partially due to the symmetry in the formulation of the integer linear program used by Sartono et al. (2015a). An integer linear program is symmetric if its variables can be permuted without changing the structure of the problem (Margot, 2010). Similarly, the mixed integer linear program formulation in the two-step approach used by Sartono et al. (2015b) to arrange a given subplot design in blocks also involves symmetry, and therefore suffers from long computation times too. In the problem of finding optimal blocking arrangements, the symmetry is caused by the fact that the blocks can be relabeled without affecting the solution quality.

To illustrate the symmetry in the problem of finding an orthogonal blocking pattern for a given orthogonal array, assume that we wish to arrange the orthogonal array $\mathbf{X}$ with 32 runs and four two-level treatment factors in Table 1 in four blocks of size eight. One arrangement of the 32 runs in four blocks of size eight is given in the column labeled $\mathbf{b}_1$, where the blocks are labeled 1, 2, 3 and 4. It can be verified that this blocking arrangement is orthogonal for the main effects: the numbers of -1s and 1s in each column of $\mathbf{X}$ are equal within each of the four blocks. In this example, there are two different symmetry problems:

- The first type of symmetry is due to the fact that we can relabel the blocks without changing the quality of the blocking pattern. For instance, whether or not the blocking pattern is orthogonal for the main effects is not affected by the labels used for the blocks. This is illustrated in Table 1 in the column labeled $\mathbf{b}_2$, which is identical to column $\mathbf{b}_1$, except for the fact that the block labels 1 and 2 have been switched. As there are four blocks in this example, there are in fact 4! possible permutations of the block labels.

- The second type of symmetry in this example is that the orthogonal array involves 16 pairs of identical runs. For instance, runs 1 and 2 are identical. When using blocking arrangement $\mathbf{b}_1$, run 1 appears in block 1, while run 2 appears in block 2. However, we can reassign runs 1 and 2 to blocks 2 and 1, respectively, without affecting the quality of the blocking arrangement. This new assignment is shown in the column named $\mathbf{b}_3$ in Table 1. As the orthogonal array in the tables contains 16 pairs of identical runs, there are 2^{16} possible ways in which the duplicated runs can be assigned to the blocks without affecting the quality of the blocking arrangement.

Table 1: An orthogonal four-factor two-level treatment design arranged in four blocks of eight runs.

Run	$\mathbf{X}$				$\mathbf{b}_1$	$\mathbf{b}_2$	$\mathbf{b}_3$
1	−1	−1	−1	−1	1	2	2
2	−1	−1	−1	−1	2	1	1
3	−1	−1	−1	1	3	3	3
4	−1	−1	−1	1	4	4	4
5	−1	−1	1	−1	3	3	3
6	−1	−1	1	−1	2	1	2
7	−1	−1	1	1	1	2	1
8	−1	−1	1	1	4	4	4
9	−1	1	−1	−1	4	4	4
10	−1	1	−1	−1	1	2	1
11	−1	1	−1	1	2	1	2
12	−1	1	−1	1	3	3	3
13	−1	1	1	−1	4	4	4
14	−1	1	1	−1	3	3	3
15	−1	1	1	1	2	1	2
16	−1	1	1	1	1	2	1
17	1	−1	−1	−1	4	4	4
18	1	−1	−1	−1	3	3	3
19	1	−1	−1	1	1	2	1
20	1	−1	−1	1	2	1	2
21	1	−1	1	−1	4	4	4
22	1	−1	1	−1	1	2	1
23	1	−1	1	1	3	3	3
24	1	−1	1	1	2	1	2
25	1	1	−1	−1	2	1	2
26	1	1	−1	−1	3	3	3
27	1	1	−1	1	1	2	1
28	1	1	−1	1	4	4	4
29	1	1	1	−1	1	2	1
30	1	1	1	−1	2	1	2
31	1	1	1	1	4	4	4
32	1	1	1	1	3	3	3

Generally, the first type of symmetry is the most problematic. This is because many screening experiments involve a limited number of experimental runs and many factors, and orthogonal arrays with small numbers of runs and many factors involve at most a few replicated runs.

As the method of Sartono et al. (2015a) is based on mixed integer linear programming, it uses a branch-and-bound algorithm to find an optimal blocking arrangement. Branch-and-bound procedures are described in detail in Wolsey (1988). When no symmetry breaking techniques are used, the branch-and-bound procedure ends up evaluating equivalent solutions that can be obtained by permuting variables when solving an optimization problem involving symmetry. This leads to prohibitively long computing times (Sherali et al., 2003). In the context of blocking orthogonal arrays, this means that the standard branch-and-bound procedure (i.e., without any form of symmetry breaking) evaluates all equivalent blocking arrangements that can be obtained by relabeling blocks and by reassigning duplicated experimental runs.

1.4 Existing literature on symmetry breaking

The problem of finding optimal blocking arrangements for orthogonal arrays to some extent resembles the job grouping problem studied by Jans and Desrosiers (2013). In the job grouping problem, we need to assign a set of jobs, each with a specific set of tool requirements, to a number of machines with limited tool capacities, so as to minimize the number of machines needed. The symmetry in this problem results from the fact that the machines are identical, and, for any given solution, the machines can be permuted to obtain an equivalent solution. To deal with the symmetry in the job grouping problem, Jans and Desrosiers (2013) use symmetry breaking techniques such as variable reduction and hierarchical ordering. These techniques have been used in the operations research literature on various occasions, for instance by Coelho and Laporte (2014) and Jans (2009) to tackle inventory-routing problems and production planning problems. Another symmetry breaking technique is to reformulate the original integer linear program and use an asymmetric representatives formulation (ARF). This kind of formulation was first proposed by Junglas (2007) and Campêlo et al. (2008) for the vertex coloring problem. A similar approach has been used by Jans et al. (2008) to group end products that use a common component. An ARF was generalized for binary clustering problems by Jans and Desrosiers (2010), and has been applied to the aforementioned job grouping problem by Jans and Desrosiers (2013). The ARF approach has also been used to develop a class representative model for pure parsimony haplotyping (Catanzaro et al., 2010) in the field of genetic data analysis. It has also been discussed by Margot (2010), who provides a good introduction to symmetry in integer linear programing (ILP). Symmetry breaking constraints have further been applied to various problems such as allocating surgery blocks (Denton et al., 2010), facility layout planning (Sherali et al., 2003), production and inventory routing (Adulyasak et al., 2014; Coelho and Laporte, 2013), workforce design (Bard and Wan, 2008), the deficiency problem for edge coloring (Altinakar et al., 2016), timetabling (Boland et al., 2008), computer clustering (Van der Gaast et al., 2014) and freight consolidation (Melo and Ribeiro, 2015).

Reformulations and adding symmetry breaking constraints to the formulation are not the only way to counter the problem of symmetry in mixed integer programming problems. Another line of research focuses on developing algorithms that dynamically break symmetry during the branch-and-bound procedure. Examples of this type of research include the orbitopal fixing approach (Kaibel and Pfetsch, 2008; Kaibel et al., 2011), isomorphism pruning (Margot, 2002, 2003; Linderoth et al., 2009), and orbital branching (Ostrowski et al., 2011, 2015). Solvers such as CPLEX and GUROBI make use of these symmetry-breaking approaches by default.

1.5 Goals and outline of this paper

The main goals of this paper are to speed up the approach of Sartono et al. (2015a) by adding symmetry breaking constraints to their formulation for finding optimal blocking arrangements of orthogonal arrays, and to develop a new formulation that does not suffer from problems of symmetry. Achieving these goals will allow future experimenters to generate orthogonal blocking arrangements for large orthogonal arrays within acceptable computing times.

A secondary goal of this paper is to investigate whether adding symmetry breaking constraints to a formulation or using alternative formulations remains useful, given that the solvers already use symmetry breaking features by default.

The remainder of the paper is organized as follows. In Section 2.1, we introduce the original problem formulation used by Sartono et al. (2015a). In Section 3, we propose several constraints that can be added to the original formulation to break the symmetry due to the blocks as well as due to the occurrence of identical runs. The constraints we propose are based on variable reduction, hierarchical ordering, and lexicographical ordering. In addition, we employ the objective perturbation technique to destroy symmetry in the orginal formulation. In Section 4, we present an asymmetric representatives formulation for the problem. Additionally, we propose several symmetry breaking constraints that can be added to the asymmetric representatives formulation to break the symmetry due to the occurrence of identical runs. In Section 5, we test the different problem formulations using a set of two-level orthogonal arrays with run sizes from 32 up to 48. Finally, in Section 6, we discuss our findings as well as some opportunities for future research.

2 Original problem formulation

2.1 Problem description

Our goal is to arrange the N runs of a two-level orthogonal array $\mathbf{X}$ involving q_1 two-level treatment factors into b blocks or groups of size N/b. We label the b blocks $1, 2, \ldots, b$. Each element of $\mathbf{X}$ is either -1 or $+1$. We denote the element of $\mathbf{X}$ in row i and column j by x_{ij}. That element is the level of the jth experimental factor in the ith run of the experiment.

Every column of $\mathbf{X}$ is a main effect contrast vector, because it is used to estimate the main effect of the corresponding experimental factor. Generally, experimenters are not only interested in estimating the factors' main effects, but also in estimating the two-factor interaction effects. This requires the calculation of the so-called two-factor interaction contrast vectors. These are the element-wise products of any two columns of $\mathbf{X}$. The $(N \times q_1(q_1-1)/2)$-dimensional matrix involving all two-factor interaction contrasts vectors is the two-factor interaction contrast matrix, which we denote by $\mathbf{W}$. We denote the number of columns of $\mathbf{W}$, $q_1(q_1-1)/2$, by q_2, and the element of $\mathbf{W}$ in row i and column j by w_{ij}.

An arrangement of the N runs of $\mathbf{X}$ in b blocks can be represented using an $N \times b$ binary assignment matrix $\mathbf{B}$ whose elements b_{ij} take the value 1 when the i-th run of $\mathbf{X}$ is assigned to the j-block, and 0 otherwise. Because we want to estimate the main effects of the experimental factors independently from the block effects, we want the binary assignment matrix $\mathbf{B}$ to be orthogonal to the matrix $\mathbf{X}$. Therefore, we want to optimize $\mathbf{B}$ subject to the constraint that all elements of the $(q_1 \times b)$-dimensional matrix product $\mathbf{X}^T\mathbf{B}$ are zero. More specifically, subject to this orthogonal blocking constraint, we want to minimize the extent to which estimates of two-factor interactions are confounded with the blocks. In mathematical terms, we want the elements of the $(q_2 \times b)$-dimensional matrix product $\mathbf{S} = \mathbf{W}^T\mathbf{B}$ to be as close to zero as possible. In the remainder of this paper, we denote the element of $\mathbf{S}$ in row i and column j by s_{ij}. That element can be calculated as $s_{ij} = \sum_{k=1}^{N} w_{ki}b_{kj}$.

To make the elements of $\mathbf{S}$ as close to zero as possible, Sartono et al. (2015a) adopt a hierarchical approach. The primary objective in that approach is to minimize the maximum absolute element of the matrix $\mathbf{S}$, which we denote by $s = \max\{|s_{ij}|, i = 1, \ldots, q_2; j = 1, \ldots, b\}$. The secondary objective is to minimize the sum of all absolute values of the elements of $\mathbf{S}$. We can calculate that sum as $\gamma = \sum_{i=1}^{q_2}\sum_{j=1}^{b}|s_{ij}|$.

2.2 Integer linear program

In order to avoid absolute values in our optimization model, Sartono et al. (2015a) use two auxiliary (non-negative) variables, s_{ij}^+ and s_{ij}^-, for each element s_{ij} of $\mathbf{S}$, so that $s_{ij} = s_{ij}^+ - s_{ij}^-$. The first of these auxiliary variables is strictly positive when $s_{ij} > 0$, and zero otherwise. The second of the auxiliary variables is strictly positive when $s_{ij} < 0$, and zero otherwise. In terms of the auxiliary variables, we can rewrite our primary objective function as

$$s = \max\{s_{ij}^+, s_{ij}^-, i = 1, \ldots, q_2; j = 1, \ldots, b\},$$

and our secondary objective function as

$$\gamma = \sum_{i=1}^{q_2}\sum_{j=1}^{b} s_{ij}^+ + \sum_{i=1}^{q_2}\sum_{j=1}^{b} s_{ij}^-,$$

with

$$s_{ij}^{+} - s_{ij}^{-} = \sum_{k=1}^{N} w_{ki} b_{kj}.$$

Since we have a primary and a secondary objective, we use pre-emptive goal programming (Ignizio, 1983). Our combined objective function can be written as

$$\min f = Ms + \sum_{i=1}^{q_2} \sum_{j=1}^{b} s_{ij}^{+} + \sum_{i=1}^{q_2} \sum_{j=1}^{b} s_{ij}^{-},$$

where M is a large number.

In summary, we use the following integer linear programming (ILP) model to assign the N runs of a given orthogonal array $\mathbf{X}$ to b blocks of size N/b:

$$\min f = Ms + \sum_{i=1}^{q_2} \sum_{j=1}^{b} s_{ij}^{+} + \sum_{i=1}^{q_2} \sum_{j=1}^{b} s_{ij}^{-} \tag{1}$$

subject to

$$\sum_{k=1}^{N} w_{ki} b_{kj} - s_{ij}^{+} + s_{ij}^{-} = 0, \quad i = 1, \ldots, q_2; j = 1, \ldots, b \tag{2}$$

$$\sum_{k=1}^{N} x_{ki} b_{kj} = 0, \quad i = 1, \ldots, q_1; j = 1, \ldots, b \tag{3}$$

$$\sum_{i=1}^{N} b_{ik} = N/b, \quad k = 1, \ldots, b \tag{4}$$

$$\sum_{k=1}^{b} b_{ik} = 1, \quad i = 1, \ldots, N \tag{5}$$

$$s_{ij}^{+} \leq s, \quad i = 1, \ldots, q_2; j = 1, \ldots, b \tag{6}$$

$$s_{ij}^{-} \leq s, \quad i = 1, \ldots, q_2; j = 1, \ldots, b \tag{7}$$

$$b_{ij} \in \{0, 1\}, \quad i = 1, \ldots, N; j = 1, \ldots, b \tag{8}$$

$$s_{ij}^{+} \geq 0, \quad i = 1, \ldots, q_2; j = 1, \ldots, b \tag{9}$$

$$s_{ij}^{-} \geq 0, \quad i = 1, \ldots, q_2; j = 1, \ldots, b. \tag{10}$$

Equation (1) in this model represents the objective function, which in a hierarchical way minimizes the confounding between the two-factor interaction contrast vectors contained within the matrix $\mathbf{W}$ and the block effects. Equation (2) ensures that either s_{ij}^{+} or s_{ij}^{-} represents the absolute value of the element s_{ij} of the matrix $\mathbf{S}$. Equation (3) ensures that the main-effect contrast vectors are orthogonal to the blocks by imposing that all the elements of the matrix product $\mathbf{X}^T\mathbf{B}$ are zero. Equations (4) and (5) indicate that every block should include N/b runs and that every run should be assigned to exactly one block. Equations (6) and (7) define the maximum absolute element of $\mathbf{S}$. Finally, Equations (8), (9), and (10) are the binary constraints for the elements of the assignment matrix $\mathbf{B}$ and the nonnegativity constraints for s_{ij}^{+} or s_{ij}^{-}.

3 Symmetry breaking approaches for the original formulation

First, we propose constraints that can be added to the original formulation to break the symmetry due to the blocks. This is the most important kind of symmetry in screening experiments with many factors. Second, we propose constraints that can be added to the original formulation to break the symmetry due to identical runs.

We compare a broad variety of symmetry breaking approaches whereas previous research generally focused on a smaller subset of approaches such as variable reduction and lexicographic ordering (Jans and Desrosiers, 2013), hierarchical ordering (Coelho and Laporte, 2014), objective perturbation (Ghoniem and Sherali, 2011), or an asymmetric representatives formulation (Campêlo et al., 2008).

3.1 Symmetry due to the blocks

3.1.1 Variable reduction

One approach to reduce the symmetry in an integer programming formulation is variable reduction, i.e. the elimination of some of the variables (Jans and Desrosiers, 2013). Since we can arbitrarily label the blocks in any blocking arrangement, we can assign the first run of the orthogonal array under consideration to any of the b blocks. To break this symmetry, we assign the first run to block 1. This is done by setting $b_{11} = 1$, which is equivalent to setting $b_{1j} = 0$, for each $j > 1$. The second run can be assigned either to the same block as the first run (when the optimal assignment requires runs 1 and 2 to be assigned to the same block), or to another block (when the optimal assignment requires runs 1 and 2 to be assigned to different blocks). In the latter case, we have the freedom to assign the second run to any of the $b-1$ remaining blocks. To break the symmetry in the assignment of the second run, we impose the constraint that the second run is either assigned to the first block or to the second block. This can be done by adding the constraint $b_{21} + b_{22} = 1$ to the original problem formulation, or equivalently, by setting $b_{2j} = 0$, for each $j > 2$. The same kind of reasoning can be used to add symmetry breaking constraints for each of the runs $3, \ldots, b-1$. This approach to symmetry breaking can be summarized using the following equation:

$$b_{ij} = 0, \quad i = 1, \ldots, b-1; j = i+1, \ldots b. \tag{11}$$

Since some of variables are set to zero, and hence removed from the optimization problem, this technique is referred to as 'variable reduction'.

3.1.2 Lexicographical ordering

Another way to break the symmetry in the original problem formulation is to impose that the blocks are ordered in a lexicographic way, based on the assignment of the runs to the b blocks. The main idea in the lexicographic ordering is that we first order the blocks based on the presence of the first run. Obviously, this ordering will not be complete and a tie will occur between all $b-1$ blocks that do not contain the first run. To deal with the tie, the presence of the second run can be used as a secondary criterion to rank the blocks. However, depending on whether or not the first and the second run are assigned to the same block, there will still be a tie between $b-1$ or $b-2$ blocks, and a tertiary criterion is needed for ranking, i.e., the presence of the third run. Continuing this line of reasoning eventually results in a complete ranking of all b blocks.

Technically, ordering the blocks based on the presence of the first run is achieved by adding the following constraints to the original problem formulation:

$$b_{11} \geq b_{12} \geq \ldots \geq b_{1b},$$

or, equivalently,

$$2^{1-1}b_{11} \geq 2^{1-1}b_{12} \geq \ldots \geq 2^{1-1}b_{1b}. \tag{12}$$

To attempt to break the tie between the $b-1$ blocks that do not contain the first run, we use the following constraint, which also takes into account the assignment of the second run:

$$2b_{11} + b_{21} \geq 2b_{12} + b_{22} \geq \ldots \geq 2b_{1b} + b_{2b},$$

or, equivalently,

$$2^{2-1}b_{11} + 2^{1-1}b_{21} \geq 2^{2-1}b_{12} + 2^{1-1}b_{22} \geq \ldots \geq 2^{2-1}b_{1b} + 2^{1-1}b_{2b}. \tag{13}$$

The appropriate choice of the coefficients of the decision variables will ensure that the first run acts as the primary criterion for ordering the blocks and the second run is the secondary criterion when there is a tie with respect to the first criterion.

Since Constraint (13) will also not result in a unique ordering of the blocks, we can also take into account the assignment of the third run. An appropriate constraint using the presence of the first, second and third run as primary, secondary and tertiary ranking criteria is given by

$$4b_{11} + 2b_{21} + b_{31} \geq 4b_{12} + 2b_{22} + b_{32} \geq \ldots \geq 4b_{1b} + 2b_{2b} + b_{3b},$$

or, equivalently,

$$2^{3-1}b_{11} + 2^{2-1}b_{21} + 2^{1-1}b_{31} \geq 2^{3-1}b_{12} + 2^{2-1}b_{22} + 2^{1-1}b_{32} \geq \ldots \geq 2^{3-1}b_{1b} + 2^{2-1}b_{2b} + 2^{1-1}b_{3b}.$$

To ensure that a unique ordering is obtained, the above constraints need to be generalized to take into account all runs when ranking the b blocks. This yields the following set of constraints:

$$\sum_{i=1}^{N} 2^{N-i} b_{i,k-1} \geq \sum_{i=1}^{N} 2^{N-i} b_{ik}, \quad k = 2, \ldots, b. \tag{14}$$

One way to interpret the constraints in Equation (14) is as follows. First, we calculate the value $\sum_{i=1}^{N} 2^{N-i} b_{ik}$ for each block k, and, next, we impose that the blocks are arranged in decreasing order of their value. The value of a block depends on the runs it contains, and the weights used for each of the N runs. In Equation (14), the weight of the ith run is 2^{N-i}. Using the weights 2^{N-i} ensures a unique ordering of all the blocks, following the lexicographic principle that the block containing the first run is ranked first, followed by the block that contains the second run, etc. The fact that the ordering is unique implies that the set of constraints in Equation (14) is ideal, as it eliminates all symmetry due to the blocks. The constraints, however, have one drawback: the weights used for the N runs can become extremely large and potentially result in numerical difficulties for large numbers of runs.

While the constraints in Equation (14) result in a unique ordering by themselves, it may be useful to add all of the lexicographic constraints in Equations (12)–(14) to the original problem formulation, instead of only those in Equation (14) itself. A concise way of writing all these constraints is as follows:

$$\sum_{i=1}^{j} 2^{j-i} b_{i,k-1} \geq \sum_{i=1}^{j} 2^{j-i} b_{ik}, \quad k = 2, \ldots, b; j = 1, \ldots, N. \tag{15}$$

It is possible to use a similar ordering logic with a set of smaller weights for the N runs, to avoid numerical problems. The ordering resulting from the use of a different set of weights will generally only be a partial ordering of the b blocks, but it will still remove some of the symmetry in the original problem formulation. One possible weight for the ith run that we will test in this paper is the weight $N - i$ instead of 2^{N-i}. This results in the following set of constraints:

$$\sum_{i=1}^{N} (N-i) b_{i,k-1} \geq \sum_{i=1}^{N} (N-i) b_{ik}, \quad k = 2, \ldots, b. \tag{16}$$

In order to see how the constraints in Equations (14) and (16) help to reduce the symmetry between the blocks, consider the three equivalent blocking arrangements labeled $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ in Table 1. The three blocking arrangements all involve four blocks of eight runs. We can now calculate the value of each block in the three blocking arrangements, using the weights 2^{N-i} and $N - i$ for the individual runs. The resulting values for the three arrangements appear in Table 2.

Table 2 shows that the weights 2^{N-i} result in a unique value for each of the blocks, no matter what blocking arrangement is considered. Importantly, the values obtained for the blocks in blocking arrangements $\mathbf{b}_2$ and $\mathbf{b}_3$ violate the constraints in Equation (14). In other words, when adding the constraints in Equation (14) to the original problem formulation, two of the three equivalent blocking arrangements in Table 1 become infeasible and will be eliminated during the branch-and-bound procedure.

The weights $N - i$ do not result in a unique value for each of the blocks. For instance, in blocking arrangement $\mathbf{b}_1$, both blocks 1 and 4 have a total weight of 125, while blocks 2 and 3 have a total weight of 123. As a consequence, it is possible to switch the labels of several blocks without changing the total weights, so that not all symmetry has been broken. Nevertheless, the constraints in Equation (16) do have added

Table 2: Total weights for the four blocks in blocking arrangements $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ in Table 1 when the weight of the ith run is 2^{N-i} (left) and $N-i$ (right).

Arr.	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4
$\mathbf{b}_1$	2,185,307,176	1,143,083,396	672,416,321	294,160,402	125	123	123	125
$\mathbf{b}_2$	1,143,083,396	2,185,307,176	672,416,321	294,160,402	123	125	123	125
$\mathbf{b}_3$	1,111,565,352	2,216,825,220	672,416,321	294,160,402	124	124	123	125

value. This is because none of the blocking arrangements $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ satisfy these constraints, so that none of these blocking arrangements will be considered in the branch-and-bound procedure for identifying an optimal solution. However, by switching blocks 2 and 4 in arrangement $\mathbf{b}_1$, we would obtain an acceptable solution under the constraints in Equation (16).

3.1.3 Hierarchical ordering

Coelho and Laporte (2014) suggest yet another type of symmetry breaking constraint, which is called hierarchical ordering and which does not involve large numbers. The idea of hierarchical ordering is that, if we assign the ith run to the kth block, then at least one of the runs $1, \ldots, i-1$ should have been assigned to block $k-1$. This kind of ordering is enforced by the following constraints:

$$b_{ik} \leq \sum_{j=1}^{i-1} b_{j,k-1}, \quad i = 2, \ldots, N; k = 2, \ldots, b \tag{17}$$

3.2 Symmetry due to replicated runs

In this section, we propose constraints to break symmetry due to the occurrence of identical runs in the orthogonal array that has to be arranged in blocks. The main idea here is that, if the first occurrence of a replicated run is assigned to block i, then the second occurrence should be assigned to one of the blocks $i, \ldots, b$. Likewise, if the second occurrence of a replicated run is assigned to a block j, then the third occurrence has to be assigned to one of the blocks $j, \ldots, b$, etc.

It is not very difficult to cast this into a linear inequality. Suppose that the runs i and j are identical and $i < j$. Then, the constraint

$$\sum_{k=1}^{b} k \; b_{ik} \leq \sum_{k=1}^{b} k \; b_{jk} \tag{18}$$

produces the desired kind of assignment of the two replicates to the blocks. To break all the symmetry due to the replicated runs of an orthogonal array, we need to add a constraint such as the one in Equation (18) for each pair of identical runs.

3.3 Perturbing the objective function

Rather than adding symmetry breaking constraints to an integer linear program to speed up the computations, Ghoniem and Sherali (2011) suggested modifying, or perturbing, the objective function. By means of this technique, we modify the objective function so that equivalent blocking arrangements end up having a different value for the objective function. Ideally, the objective function is modified so that only one of all the equivalent blocking arrangements results in the optimal objective function value. The technique proposed by Ghoniem and Sherali (2011) requires adding a weighted sum of expressions used in symmetry breaking constraints to the original objective function.

Because their approach is based on the symmetry breaking constraints in Equation (14), the values $\sum_{i=1}^{N} 2^{N-i} b_{ik}$ calculated for each block k are added to the objective function, with appropriate weights. We ensure that the weights increase with the block label $(1, \ldots, b)$, so that they are compatible with the original 'larger than' constraint in Equation (14). By using a larger weight for larger block labels, there is an incentive to minimize the value $\sum_{i=1}^{N} 2^{N-i} b_{ik}$ for the larger block labels rather than for the smaller block numbers.

This implies that blocking arrangements that are equivalent in terms of the original objective function are no longer equivalent in terms of the newly modified, or perturbed, objective function. Consequently, this approach breaks the symmetry due to the blocks.

Obviously, the original objective function, which aims to minimize the confounding of the two-factor interactions with the blocks, should receive a higher priority in the newly modified objective function than the perturbation added to break the symmetry between the blocking arrangements. For this reason, the original objective function should get a sufficiently large weight μ in the newly modified objective function. In the context of our search for optimal blocking arrangement, the modified objective function is

$$\min f = \mu(Ms + \sum_{i=1}^{q_2}\sum_{j=1}^{b} s_{ij}^{+} + \sum_{i=1}^{q_2}\sum_{j=1}^{b} s_{ij}^{-}) + (\sum_{k=1}^{b} \frac{1}{b-k+1} \sum_{i=1}^{N} 2^{N-i} b_{ik}), \tag{19}$$

$$\tag{20}$$

where

$$\mu = 2^N \sum_{k=1}^{b} \frac{1}{b-k+1}.$$

Ghoniem and Sherali (2011) obtained the best results for their optimization problem by combining the objective function perturbation with the addition of the original symmetry breaking constraints. For this reason, in our comparisons, we also combine the perturbed objective function in Equation (19) with the lexicographic ordering constraints in Equation (14) when searching for optimal blocking arrangements.

4 An alternative problem formulation

In this section, we reformulate the original optimization problem by means of an asymmetric representatives formulation (ARF), which was introduced in the literature by Campêlo et al. (2008) for the vertex coloring problem, and later used by Jans and Desrosiers (2013) for the job grouping problem. The ARF can be readily implemented to avoid symmetry between the blocks in our problem of finding optimal blocking arrangements. We also show how symmetry breaking constraints can be added to the ARF to deal with identical runs.

4.1 The asymmetric representatives formulation

The major source of symmetry in the original problem formulation is the fact that b blocks or groups of runs need to be identified and that the groups have labels that can be permuted. An alternative formulation, the asymmetric representatives formulation, keeps track of which runs appear in the same block too, but it uses a label for each block that depends on the runs it contains. More specifically, a block is identified or labeled by its lowest-indexed run. As a result, it is no longer possible to swap the labels of the blocks.

We illustrate this idea using a small example. Suppose that we wish to arrange six runs, labeled $1, \ldots, 6$, in three blocks of size two. One possible blocking arrangement groups runs 1 and 3, runs 2 and 5, and runs 4 and 6. Because the lowest index for a run in the first block is 1, we label the first block as 1. Likewise, the lowest index for a run in the second block is 2, and, therefore, we label the second block as 2. Finally, the lowest index in the third block is 4. Therefore, we label the third block as 4. To uniquely identify this blocking pattern, all we need to know is that

1. The blocks' labels are 1, 2 and 4;
2. Run 3 appears in the same block as run 1;
3. Run 5 appears in the same block as run 2;
4. Run 6 appears in the same block as run 4.

Now, the challenge is to find a mathematical formulation that is based on this kind of information. Building on the work of Campêlo et al. (2008) and Jans and Desrosiers (2013), we use $N(N+1)/2$ binary decision variables v_{ij}, where the first index i runs from 1 to N and the second index j runs from i to N.

Table 3: Values for the decision variables v_{ij} in the example in Section 4.1. The symbol '-' indicates a variable that is not used.

	j					
i	1	2	3	4	5	6
1	1	-	-	-	-	-
2	0	1	-	-	-	-
3	1	0	0	-	-	-
4	0	0	0	1	-	-
5	0	1	0	0	0	-
6	0	0	0	1	0	0

A variable v_{ij} takes the value one if run i and run j are assigned to the same block and if j is the smallest index in that block. Otherwise, v_{ij} takes the value zero. In our small example, the variables v_{11}, v_{22} and v_{44} as well as v_{31}, v_{52}, and v_{64} take the value one, while all other variables, including v_{33}, v_{55} and v_{66}, take the value zero. Table 3 provides an overview of all decision variables and their values for the example. Note that v_{66} necessarily equals zero, because it is impossible for the 6th run to be the run with the smallest index in any block of size two.

The asymmetric representatives formulation is based on the variables v_{ij}. We call run j the block identifier of run i. Whenever $v_{ii} = 1$ for a particular run i, this indicates that there exists a block in which run i has the smallest index. Since, in general, we need b blocks, there should be exactly b different variables v_{ii} which take the value one, and $N - b$ variables v_{ii} which take the value zero. We do not need any other decision variables to ensure that b blocks are utilized. Note that all variables v_{ii} for which $i > N - N/b + 1$ must be zero. This is because any block with identifier i has to contain N/b runs selected from the set $\{i, \ldots, N\}$. This is only possible if the cardinality of the set $\{i, \ldots, N\}$ is at least N/b, i.e., if $i \leq N - N/b + 1$. This allows us to limit the number of decision variables and constraints in the ARF formulation. In total, there are $N(N+1)/2 - (N/b)((N/b)-1)/2$ decision variables v_{ij} which can be nonzero.

As before, the objective function minimizes the maximum absolute element of the matrix $\mathbf{S}$ which quantifies the confounding between the two-factor interaction contrast vectors and the blocks, followed by the sum of all absolute elements of $\mathbf{S}$. Unlike the original problem formulation, the ARF formulation does not involve the $N \times b$ binary elements b_{ij} of the matrix $\mathbf{B}$, which assigns the runs to the blocks. Instead, it involves the binary variables v_{ik}. This has a substantial impact on the formulation of the constraints. In the new formulation, the decision variables v_{ik} have to be used to determine the elements of $\mathbf{S}$, because these variables indicate whether or not runs have been assigned to a block involving run k as the run with the smallest index. If a certain block identifier k is not used ($v_{kk} = 0$), we cannot assign any run j to a block with label k. Thus, the elements of $\mathbf{S}$ related to that block label will be zeros. If a certain block identifier k is used ($v_{kk} = 1$), these decision variables play a role similar to that of the variables b_{ik} equal to one in the original formulation, and allow the calculation of the elements of $\mathbf{S}$ corresponding to the block with identifier k. These elements can be positive or negative, and are calculated as

$$s_{ij} = \sum_{k=j}^{N} w_{ki} v_{kj}, \quad i = 1, \ldots, q_2; j = 1, \ldots, N - N/b + 1.$$

We denote their absolute values by s_{ij}^{+} or by s_{ij}^{-}, depending on whether they are positive or negative, as in the original problem formulation. In a similar fashion, we need to calculate the elements of the matrix product $\mathbf{X}^T\mathbf{B}$ to check the orthogonality of the blocking arrangement for the main effect contrast vectors. The elements of that matrix are computed as

$$\sum_{k=j}^{N} x_{ki} v_{kj}, \quad i = 1, \ldots, q_1; j = 1, \ldots, N - N/b + 1,$$

and they should equal zero for any orthogonal blocking arrangement.

The complete ARF formulation for the problem of finding an optimal blocking arrangment for the N

runs of an orthogonal array is as follows:

$$\min f = Ms + \sum_{i=1}^{q_2} \sum_{j=1}^{N-N/b+1} s_{ij}^{+} + \sum_{i=1}^{q_2} \sum_{j=1}^{N-N/b+1} s_{ij}^{-} \tag{21}$$

subject to

$$\sum_{k=j}^{N} w_{ki} v_{kj} - s_{ij}^{+} + s_{ij}^{-} = 0, \quad i = 1, \ldots, q_2; j = 1, \ldots, N - N/b + 1 \tag{22}$$

$$\sum_{k=j}^{N} x_{ki} v_{kj} = 0, \quad i = 1, \ldots, q_1; j = 1, \ldots, N - N/b + 1 \tag{23}$$

$$\sum_{i=k}^{N} v_{ik} = (N/b) \times v_{kk}, \quad k = 1, \ldots, N - N/b + 1 \tag{24}$$

$$\sum_{k=1}^{\min(i, N-N/b+1)} v_{ik} = 1, \quad i = 1, \ldots, N \tag{25}$$

$$\sum_{i=1}^{N-N/b+1} v_{ii} = b, \tag{26}$$

$$v_{ik} \leq v_{kk}, \quad k = 1, \ldots, N - N/b + 1, i = k + 1, \ldots, N, \tag{27}$$

$$s_{ij}^{+} \leq s, \quad i = 1, \ldots, q_2; j = 1, \ldots, N - N/b + 1, \tag{28}$$

$$s_{ij}^{-} \leq s, \quad i = 1, \ldots, q_2; j = 1, \ldots, N - N/b + 1, \tag{29}$$

$$v_{ij} \in \{0, 1\}, \quad i = 1, \ldots, N; j = 1, \ldots, N - N/b + 1, \tag{30}$$

$$s_{ij}^{+} \geq 0, \quad i = 1, \ldots, q_2; j = 1, \ldots, N - N/b + 1, \tag{31}$$

$$s_{ij}^{-} \geq 0, \quad i = 1, \ldots, q_2; j = 1, \ldots, N - N/b + 1. \tag{32}$$

Constraint (22) identifies the elements of the matrix $\mathbf{S}$ as well as the auxiliary variables s_{ij}^{+} and s_{ij}^{-}. Constraint (23) ensures that the blocking arrangement is orthogonal for the main effect contrast vectors. Constraint (24) ensures that the number of runs assigned to the block with identifier k is N/b if that block is used. More specifically, that constraint ensures that, if $v_{kk} = 1$ (i.e., if there is a block with run k as the run with the smallest index), then that block involves N/b runs in total. Constraint (25) ensures that every run is assigned to exactly one block. Constraint (26) guarantees that the total number of blocks used equals b. Constraint (27) ensures that no runs are assigned to a block with identifier k if that block is not used (i.e., if $v_{kk} = 0$). Constraints (28) and (29) make sure that s reflects the maximum absolute element of $\mathbf{S}$. Finally, Constraints (30), (31), and (32) indicate which decision variables are binary and which are positive integer values.

4.2 Symmetry due to replicated runs

The ARF formulation does not suffer from symmetry due to the blocks, because the blocks cannot be relabeled. However, it may still suffer from symmetry due to the presence of identical runs in an orthogonal array. Therefore, in this section, we propose a constraint that can be added to the ARF to break the symmetry due to identical runs. The main idea now is that, if the first occurrence of a replicated run, say run i, is assigned to a block with identifier k_i (in which case $v_{ik_i} = 1$) and the second occurrence of the replicated run, say run j, is assigned to a block with identifier k_j (in which case $v_{jk_j} = 1$), then k_i should be smaller than or equal to k_j. Likewise, if the second occurrence of a replicated run is assigned to a block with identifier k_j, then the third occurrence has to be assigned to a block with an identifier larger than or

equal to k_j, etc. This is achieved by adding the symmetry breaking constraint

$$\sum_{k_i=1}^{\min(i,N-N/b+1)} k_i \; v_{ik_i} \leq \sum_{k_j=1}^{\min(j,N-N/b+1)} k_j \; v_{jk_j}, \tag{33}$$

to the ARF for each pair of replicated runs i and j, where $i < j$.

5 Computational experiments

In this section, we report the results of several computational experiments intended to find the best formulation for identifying optimal blocking patterns in terms of computing time and objective function value. For the comparison of all the approaches described above, we first focus on the blocking of 32-run orthogonal arrays. After eliminating the least fruitful approaches, we turn our attention to 40-run and 48-run orthogonal arrays. The orthogonal arrays we consider are strength-3 orthogonal arrays listed by Schoen and Mee (2012). These orthogonal arrays have excellent properties in terms of main effects and interaction effects estimation.

5.1 Preliminaries

For the 32-run and 40-run designs, we performed all tests under Windows 7 (64 bits) on a 3.4GHz Intel Core i7-3770 with a 16 GB internal memory. For the 48-run designs, we performed all tests under Windows 7 (64 bits) on a 3.04GHz Intel Xeion X5667 with an internal memory of 12 GB. For all computational experiments described in Sections 5.2–5.4, we used MATLAB 2012b along with CPLEX version 12.5. We used the default settings of CPLEX, including the default setting for symmetry breaking. The default relative mixed integer programming (MIP) gap tolerance is 0.01%, meaning that the CPLEX solver reports a solution as optimal as soon as the relative MIP gap is smaller than or equal to 0.01%. In Section 5.5, we investigate the impact of the various CPLEX settings when it comes to symmetry breaking, as well as the difference between the CPLEX solver and the GUROBI solver.

For the value of M in the various formulations of our problem, we used $N \times q_2 + 1$. This is because the maximum sum of all elements in $\mathbf{S}$ is $N \times q_2$. This is unlike Sartono et al. (2015a), who used a value of $10,000$.

In our discussion of the computational results, we refer to the original formulation as OF. We refer to the variable reduction technique as VR, to the lexicographic ordering technique as LEX, to the hierarchical ordering technique as HO, and to objective perturbation as OBJ. For each of the LEX approaches, we add the relevant equation between parentheses. Whenever we use symmetry breaking constraints for identical runs, we refer to it as IR.

5.2 Comparing different formulations using 32-run two-level designs

In this section, we test the performance of the OF with and without symmetry breaking constraints as well as the ARF when it comes to blocking the 15 32-run orthogonal arrays listed by Schoen and Mee (2012). We initially focus on the symmetry due to the blocks since, except for the four-factor array shown in Table 1, no strength-3 32-run orthogonal arrays with replicated runs are reported by Schoen and Mee (2012). We start by adding a single type of symmetry breaking constraint to the OF. Next, we select the best performing types of symmetry breaking constraints and test feasible combinations of these.

For our tests with 32-run orthogonal arrays, we used a computing time limit of two hours. We considered blocking arrangements with four blocks of eight runs and eight blocks of four runs.

5.2.1 Using a single type of symmetry breaking constraint

We summarize our initial results for 32 runs in Table 4. The first column indicates the various problem formulations considered. The second column shows the average computing times in seconds required to arrange the 15 32-run orthogonal arrays in four blocks of eight. For this scenario, all optimization problems, with or without symmetry breaking constraints, were solved to optimality well within the computing time

Table 4: Average computing times for finding optimal arrangements of 15 32-run orthogonal arrays in four and eight blocks using different problem formulations.

Formulation	4 blocks	8 blocks		
	CPU-T(s)	Optimal solutions	CPU-T(s)	Gap(%)
OF	2.31	12/12	583.18	0.00%
		0/3	7200.17	0.61%
OF-VR	2.08	12/12	708.02	0.00%
		3/3	4083.51	0.00%
OF-LEX(14)	2.69	12/12	9.60	0.00%
		3/3	33.30	0.00%
OF-LEX(15)	3.19	12/12	12.22	0.00%
		3/3	49.90	0.00%
OF-LEX(16)	7.83	6/12	4887.02	1.58%
		0/3	7204.46	4.19%
OF-HO	11.65	12/12	1593.82	0.01%
		2/3	5929.92	0.12%
OBJ	2.50	12/12	11.98	0.00%
		3/3	25.28	0.00%
ARF	8.72	12/12	0.21	0.00%
		3/3	0.16	0.00%

limit. Therefore, the gaps between our final solutions and the best lower bounds obtained are all zero. The next three columns in Table 4 are all related to the arrangements with eight blocks of four runs. For each formulation, two lines of results are provided in these columns. The upper line shows average results for the 12 'easy' problems, for which the OF was solved to optimality within the computing time limit. The lower line shows average results for the three 'difficult' problems, for which the OF could not be solved to optimality within two hours. The first of the three columns for eight blocks indicates what fraction of the 12 easy problems could be solved to optimality and what fraction of the three hard problems could be solved to optimality, by any given formulation. The second of the three columns for eight blocks shows the average computing times for the easy and hard problems. The third column shows the average gap between the final solution and the best lower bound obtained.

For arranging the 15 orthogonal arrays in four blocks, all eight problem formulations in Table 4 were solved to optimality within 12 seconds. The formulations which required the smallest average computing times (2.31 and 2.08 seconds) were the OF and the OF in combination with the VR approach (OF-VR), followed by the formulation with perturbed objective function (OBJ) (2.50 seconds). The OF in combination with the hierarchical ordering resulted in the longest computing time (11.65 seconds). The ARF required 8.72 seconds, on average, to be solved to optimality. We observe that these problems are easy considering the computing time required using the OF, and symmetry breaking techniques cannot substantially improve that computing time. This is in line with other results from the literature, which indicate that adding additional symmetry breaking (on top of the default symmetry breaking used within the solver) does not substantially improve or even slows down the computation times for easy problems (Jans, 2009).

For arranging the 15 orthogonal arrays in eight blocks, we obtain a totally different picture. This time, the OF was able to solve only 12 of the 15 blocking problems to optimality within the computing time limit of two hours. For the remaining three problems, the OF resulted in an average gap of 0.61% between the best solution found and the best lower bound. The 12 problems that were solved to optimality by the OF involved the smallest numbers of factors (4–12), while the three other problems involved the largest numbers of factors (14, 15 and 16). Interestingly, five of the seven symmetry breaking approaches in Table 4 do allow us to solve the three difficult problems to optimality within the computing time limit. More specifically, the VR approach, the LEX approach based on Equation (14), the LEX approach based on Equation (15), the OBJ approach and the ARF approach converge to optimality well within the computing time limit. Of these five approaches, the VR approach is by far the slowest. The ARF approach is the fastest: it is able to solve both the easy and the difficult problems within one second of computing time. The ARF was able to solve these problem instances more than 2000 times faster than the OF. The LEX approach based on

Table 5: Average computing times for finding optimal arrangements of 15 32-run orthogonal arrays in four and eight blocks by combining two types of symmetry breaking constraints.

Formulation	4 blocks	8 blocks		
	CPU-T(s)	Optimal solutions	CPU-T(s)	Gap(%)
OF-VR-LEX(14)	2.48	12/12	16.28	0.00%
		3/3	57.00	0.00%
OF-VR-LEX(15)	3.23	12/12	12.02	0.00%
		3/3	106.70	0.00%
OF-VR-HO	2.51	12/12	8.06	0.00%
		3/3	27.54	0.00%
OF-LEX(14)-HO	3.07	12/12	9.02	0.00%
		3/3	33.23	0.00%
OF-LEX(15)-HO	2.70	12/12	12.93	0.00%
		3/3	21.50	0.00%

Equation (16) is the slowest of all seven symmetry breaking approaches for the easy problems. It is unable to solve six of the 12 easy problems to optimality and it is also not able to solve any difficult problem to optimality within two hours. The HO approach solved all easy problems to optimality, and two of the three difficult problems. Its computing time is very large compared to those of the ARF approach, the OBJ approach, the LEX approach based on Equation (14) and the LEX approach based on Equation (15).

Note that, for the easy problems, the VR approach is slower than the OF, but this is compensated by the fact that it is able to solve the three difficult problems with an average computing time of about 4084 seconds. Unlike the VR approach, the LEX approach based on Equation (14), the LEX approach based on Equation (15), the OBJ approach and the ARF approach all produced a substantial reduction in computing time, both for the easy problems and for the difficult problems. The HO approach is similar to the VR approach in terms of computing time, but it allowed only two of the three difficult problems to be solved to optimality.

5.2.2 Combining two types of symmetry breaking constraints

Our computational tests involving a single type of symmetry breaking constraint showed that all but one approach (the LEX approach involving the constraints in Equation (16)) have at least some added value. In this section, we investigate five combinations of two types of constraints. More specifically, we test the combinations VR-LEX, VR-HO, and LEX-HO. As there are two valuable variants of the LEX approach (involving the constraints in Equations (14) and (15)), we have two variants of the combination VR-LEX and two variants of the combination LEX-HO. The performance of the formulations obtained by combining two types of symmetry breaking constraints is summarized in Table 5, which has the same structure as Table 4.

When it comes to arranging the 32-run orthogonal arrays in four blocks of eight runs, the five combinations of two symmetry breaking methods perform similarly. All 15 problems were solved to optimality for that number of blocks, within four seconds. For this problem, the OF in combination with the VR approach is faster than the various combined approaches in Table 5.

For finding arrangements in eight blocks of four runs, the combined approaches in Table 5 all outperform the OF, both in terms of computing time and in terms of solution quality. The combined approaches are all able to solve the three difficult problems to optimality, unlike the OF, the OF-LEX(16) approach and the OF-HO approach in Table 4, in a limited amount of time. A remarkable result is that, separately, the VR and HO techniques resulted in long computing times for finding arrangements in eight blocks, but combining them with each other or with the LEX approach leads to a symmetry breaking approach that is competitive with the others in Table 5. We note that, for the blocking arrangements involving eight blocks, the ARF is still substantially faster, by at least a factor of 38, than any of the combined approaches. The combination of either the VR approach or the HO approach with the LEX(14) approach does not lead to substantial gains in average computing times compared to using the LEX(14) approach separately. For the LEX(15) approach, the combination with the HO approach does improve the average computing time, when considering the

three difficult problems.

5.3 Comparing different formulations using 40-run two-level designs

In our computational experiments involving 32-run orthogonal arrays, the OF-LEX(14), OF-LEX(15), ARF and OBJ approaches performed well. Also, the combined approaches exhibited a good performance. In this section, we compare the OF-LEX(14), OF-LEX(15), ARF, OBJ, OF-VR-HO and OF-LEX(15)-HO approaches to the OF for the 18 40-run orthogonal arrays recommended by Schoen and Mee (2012). These arrays are the best designs in terms of the G- and G_2- aberration criteria. For the 40-run arrays, we also used a computing time limit of two hours. Of the five combined approaches, we only consider the OF-VR-HO and OF-LEX(15)-HO ones in this section, because the OF-VR-LEX(14), OF-VR-LEX(15) and OF-LEX(14)-HO approaches did not perform substantially better than the simpler OF-LEX(14) and OF-LEX(15) approaches for the 32-run designs.

For two-level orthogonal arrays involving 40 runs, there are three types of possible orthogonal blocking arrangements. The first type of blocking arrangement involves four blocks of ten runs. The second type involves five blocks of eight runs, and the third type involves ten blocks of four runs. The set of 18 40-run arrays we consider here contains six arrays with replicated runs. Therefore, these six arrays allow us to test the added value of the constraints in Equations (18) and (33) to break the symmetry due to replicated runs in the OF and in the ARF.

5.3.1 Symmetry due to the blocks

In this section, we compare seven different formulations for the 18 40-run orthogonal arrays. The seven formulations are listed in the first column of Table 6, which has a structure similar to that of Table 4.

When applying the OF to the 18 40-run orthogonal arrays with the intention of finding blocking arrangements with four blocks of ten runs, 15 of the resulting problems could be solved to optimality within the computing time limit of two hours. As a result, for the scenario with four blocks, there are 15 easy problems and three difficult problems. The hard problems involve 7, 19 and 20 factors. Remarkably, none of the symmetry breaking approaches outperforms the OF in terms of the number of optimal solutions produced or in terms of the computing time required. Five of the six symmetry breaking approaches in Table 6 even fail to solve all easy problems to optimality. The only symmetry breaking approach that solves all easy problems combines the VR technique and the HO technique. The average computing time required by that approach for the easy problems (451.98 seconds) is only a little bit larger than that required by the OF (423.75 seconds). For the difficult problems, the gap between the best solution obtained and the best lower bound is slightly smaller for the symmetry breaking approach combining the VR and HO techniques (0.87%), the OF-LEX(14) approach (0.90%), the OF-LEX(15) approach (0.90%) and the OF-LEX(15)-HO approach (also 0.90%) than for the OF (1.02%). In terms of computing time and in terms of the gap, the ARF approach is the least attractive approach for the scenario with four blocks, followed by the OBJ approach.

When applying the OF to obtain arrangements of the 40-run orthogonal arrays in five blocks of ten runs, 14 of the 18 resulting problems can be solved to optimality within the computing time limit. The average required computing time for these easy problems is 920.01 seconds. This time, all symmetry breaking approaches other than the ARF are able to solve all easy problems with an average computing time smaller than that for the OF. The fastest approach is the one combining the VR and HO approaches (568.90 seconds). For the four difficult problems (involving 17, 18, 19 and 20 factors), the symmetry breaking approaches other than the ARF lead to a small reduction in the average gap. The smallest gap of 0.35% is obtained by the combined VR-HO approach, while the gap for the OF is 0.49%. The ARF produces an average gap of about the same size as the OF.

Finally, when applying the OF to find arrangements of the 18 40-run orthogonal arrays in ten blocks of four runs, only eight of the resulting optimization problems can be solved to optimality within the computing time limit. The average required computing time for these eight cases is 465.78 seconds when using the OF. When using symmetry breaking constraints, however, this average computing time can be reduced substantially. The largest reduction in average computing time is achieved by the ARF approach, which requires only 0.58 seconds for the eight easy problems. Besides reducing the computing time for the easy problems involving ten blocks, the symmetry breaking approaches also allow the ten difficult problems (involving 11–20 factors) with ten blocks to be solved to optimality. For the ARF approach, the average

computing time required to do so is only 7.71 seconds. The average computing time for the other symmetry breaking approaches is larger than that of the ARF by several orders of magnitude.

5.3.2 Symmetry due to identical runs

In this section, we focus on the six 40-run orthogonal arrays that involve replicated runs. These arrays involve 4, 5, 7, 8, 9 and 10 factors and 16, 30, 38, 38, 38, and 38 distinct runs (out of a total of 40 runs), respectively. These designs have at least two pairs of replicated runs. The main goal of this section is to test whether adding the symmetry breaking constraints for identical runs to the OF, the ARF, and some of the best formulations involving constraints to break the symmetry due to the blocks (i.e., the OF-VR-HO and OF-LEX(15)-HO approaches) leads to a reduction of the computing times. In total, we test and compare four different pairs of formulations: OF versus OF-IR, OF-VR-HO versus OF-VR-HO-IR, OF-LEX(15)-HO versus OF-LEX(15)-HO-IR, and ARF versus ARF-IR. The results are summarized in Table 7.

One key result is that adding symmetry breaking constraints for identical runs to the OF leads to a major inflation of the computing time. For instance, when looking for arrangements involving ten blocks of four runs, adding the constraints implies that, for five of the six arrays with replicated runs, the OF could no longer be solved to optimality within two hours of computing time, even though the plain OF could be solved to optimality within the computing time limit of two hours for these cases. In none of the cases, adding the symmetry breaking constraints for identical runs to the OF led to a substantial reduction in computing time.

Adding symmetry breaking constraints for identical runs to the ARF does lead to a reduction in computing time for nine of the 12 arrangements with four or five blocks. For example, when looking for arrangements involving four blocks of ten runs, adding the constraints implies that, for four of the six arrays with replicated runs, the computing time of the ARF decreases substantially. The average computing time for four blocks goes down from 2436.04 seconds to 1919.30 seconds. For arrangements with five blocks, adding symmetry breaking constraints for identical runs to the ARF leads to a reduction in computing time for all cases except the one involving seven factors. For arrangements with ten blocks, adding symmetry breaking constraints for identical runs to the ARF increases the computing time for every individual design, but the computing times remain extremely small in each case.

Finally, combining constraints for breaking the symmetry due to the blocks and constraints for breaking the symmetry due to identical runs leads to a reduction in computing time in some cases. The combined formulation OF-VR-HO-IR reduces the computing time for eight of the 18 cases, while the combined formulation OF-LEX(15)-HO-IR reduces the computing time for 14 of the 18 cases. Overall, the OF, OF-VR-HO and ARF-IR formulations are the best when arranging designs with replicated runs in four blocks, five blocks and ten blocks, respectively.

5.4 Comparing different formulations using 48-run two-level designs

In this section, we compare the best symmetry breaking approaches from Section 5.2 for the 31 48-run arrays recommended by Schoen and Mee (2012). For two-level experimental designs involving 48 runs, there are four types of possible orthogonal blocking arrangements. These blocking arrangements involve 4, 6, 8 and 12 blocks. For the 48-run arrays, we set the computing time limit to four hours. Because the constraints in the LEX(14) and LEX(15) approaches might lead to numerical problems (due to the large weights needed when solving problems with 48 runs), we only use the first 40 runs of each orthogonal array when implementing these symmetry breaking approaches here. In doing so, we adopt the same approach as Jans and Desrosiers (2013).

5.4.1 Symmetry due to the blocks

In this section, we apply the six best formulations from Section 5.2 to deal with the symmetry due to the blocks to 48-run designs. The results are summarized in Tables 8 and 9. The structure of these tables is similar to that of the tables for 32- and 40-run arrays.

When using the OF to arrange the 31 48-run arrays in four blocks of 12 runs, 26 of the 31 resulting optimization problems could be solved to optimality within the computing time limit of four hours. The average computing time required was 527.21 seconds. The average gap for the remaining five problems,

Table 6: Average computing times for finding optimal arrangements of 18 40-run orthogonal arrays in four, five and ten blocks with different formulations.

Formulation	4 blocks			5 blocks			10 blocks		
	Optimal solutions	CPU-T(s)	Gap(%)	Optimal solutions	CPU-T(s)	Gap(%)	Optimal solutions	CPU-T(s)	Gap(%)
OF	15/15	423.75	0.00%	14/14	920.01	0.01%	8/8	465.78	0.01%
	0/3	7200.10	1.02%	0/4	7202.01	0.49%	0/10	7205.20	2.86%
OF-LEX(14)	13/15	1188.33	0.37%	14/14	594.75	0.01%	8/8	17.39	0.00%
	0/3	7200.39	0.90%	0/4	7201.12	0.39%	10/10	1781.55	0.01%
OF-LEX(15)	13/15	1229.99	0.30%	14/14	755.69	0.01%	8/8	12.89	0.00%
	0/3	7200.28	0.90%	0/4	7200.34	0.39%	10/10	939.18	0.00%
ARF	12/15	1508.72	0.47%	12/14	1617.47	0.04%	8/8	0.58	0.00%
	0/3	7201.66	19.38%	0/4	7202.85	0.53%	10/10	7.71	0.00%
OF-LEX(15)-HO	13/15	1234.11	0.46%	14/14	652.06	0.01%	8/8	20.67	0.00%
	0/3	7200.63	0.90%	0/4	7202.22	0.38%	10/10	1273.54	0.00%
OF-VR-HO	15/15	451.98	0.00%	14/14	568.90	0.01%	8/8	8.20	0.00%
	0/3	7200.14	0.87%	0/4	7200.25	0.35%	10/10	1104.91	0.00%
OBJ	13/15	1410.64	0.58%	14/14	744.40	0.01%	8/8	14.34	0.01%
	0/3	7200.12	11.55%	0/4	7200.34	0.39%	10/10	746.80	0.01%

Table 7: Computing times for finding optimal blocking arrangements of 40-run orthogonal arrays with replicated runs.

Blocks	Factors	OF	OF-IR	OF-VR-HO	OF-VR-HO-IR	OF-LEX(15)-HO	OF-LEX(15)-HO-IR	ARF	ARF-IR
4	4	15.79	154.55	136.78	46.24	7200.09	69.84	154.22	86.00
	5	777.74	7200.01	2022.04	2394.91	7200.21	2915.28	7200.20	4164.40
	7	6.13	59.92	6.49	5.46	8.39	9.52	33.67	45.61
	8	0.78	2.78	0.64	1.40	1.45	1.06	15.29	12.29
	9	1.50	1.14	0.83	0.65	0.72	0.42	12.18	7.41
	10	649.74	7200.01	2829.03	2918.98	1712.61	3127.73	7200.66	7200.10
	average	241.95	2436.40	832.63	894.61	2687.24	1020.64	2436.04	1919.30
5	4	0.06	0.06	0.05	0.08	0.37	0.14	1.58	0.31
	5	0.08	0.06	0.08	0.08	0.33	0.11	1.86	0.41
	7	22.28	586.44	24.58	20.31	42.71	38.36	126.00	236.51
	8	265.53	3186.99	142.46	205.02	220.88	254.72	642.30	432.82
	9	24.24	386.62	17.75	19.72	27.55	21.05	213.77	166.50
	10	47.52	972.68	20.56	58.45	27.57	76.32	451.22	423.96
	average	59.95	855.48	34.25	50.61	53.24	65.12	239.45	210.09
10	4	4.28	6445.20	2.37	1.69	12.60	3.65	0.22	1.01
	5	421.34	7202.43	3.73	6.13	35.72	7.07	0.33	0.62
	7	1942.45	7204.24	13.65	10.31	31.65	25.52	0.91	1.26
	8	201.24	7204.91	4.82	3.60	10.33	7.94	0.62	1.05
	9	127.09	7200.63	2.68	2.54	7.02	3.88	0.53	0.84
	10	384.22	7200.62	3.78	5.15	11.86	6.04	0.20	0.39
	average	513.44	7076.34	5.17	4.90	18.20	9.02	0.47	0.86

involving 12, 13, 14, 23 and 24 factors, was 0.56%. Using lexicographical ordering by means of the constraints in Equation (14) is a little bit better, since that approach allowed us to solve one of the difficult problems to optimality. That approach also allowed us to solve all 26 easy problems to optimality, with an average computing time of 498.41 seconds. The other lexicographical approach, LEX(15), failed to solve one of the easy problems to optimality. Of the two combined approaches tested, the OF-LEX(15)-HO performed best, as it was able to solve 27 of the 31 problems to optimality. However, it did not outperform the OF-LEX(14) approach (neither in terms of the number of problems solved to optimality, nor in terms of computing time). The ARF approach was the poorest approach for the scenario with four blocks, as it was able to solve only 23 of the 31 problems to optimality. The fastest approach for the easy problems was the OF-VR-HO approach, with a computing time of 399.68 seconds on average.

Using the OF approach to find arrangements with six blocks allowed us to solve 13 of the 31 problems to optimality, with an average computing time of 3061.39 seconds. The average gap for the other 18 cases (involving 8, 9, and 14–24 factors) was 1.19%. Of the symmetry breaking approaches tested, the OF-VR-HO performed best: it produced optimal solutions for all easy problems in the shortest computing time (1518.56 seconds), and it was able to solve five of the 18 difficult problems to optimality. The OF-LEX(14) approach managed to solve four of the difficult problems, while the OF-LEX(15) approach, the ARF approach and the OF-LEX(15)-HO approach only allowed two of these problems to be solved. The OF-OBJ approach allowed us to solve five difficult problems to optimality, but failed to solve two easy problems. For finding arrangements with six blocks, the ARF approach performed better than for finding arrangements involving four blocks, relative to the other approaches. Its computing time is, however, still the longest of all approaches listed in Table 8, for six blocks.

For eight blocks of six runs, the OF allowed us to solve only nine of the 31 problems to optimality within the computing time limit, using an average computing time of 885.76 seconds. For three of these nine problems (involving 11, 12 and 13 factors), the OF returned a message of infeasibility within one minute of computing time. For the remaining 22 difficult problems (involving 4–6, 8–10, 14–24 factors), the average gap was 4.23%. None of the symmetry breaking approaches managed to solve a substantial number of difficult problems to optimality. Only the ARF approach allowed the solution of two of these problems. The other approaches allowed the solution of only one difficult problem. For the easy problems, all symmetry breaking approaches are faster than the OF. The ARF approach has the smallest computing time for these problems (54.30 seconds).

Finally, for 12 blocks of four runs, six problems were solved to optimality by the OF within the computing time limit. The average computing time required to do so was 6976.54 seconds. The ARF approach performed extremely well for the scenario with 12 blocks: it was able to solve all 31 problems to optimality, using a very short average computing time (less than 30 seconds). The symmetry breaking approaches based on the OF all performed better than the plain OF, but worse than the ARF. More specifically, they were all able to solve the six easy problems to optimality and 11 or 13 of the 25 difficult problems (involving 5–24 factors), and they all required similar average computing times. It should be noted, however, that the ARF is approximately 2 orders of magnitude faster than the best alternative symmetry breaking approach for 12 blocks of four runs.

5.4.2 Symmetry due to identical runs

For the three 48-run orthogonal arrays involving replicated runs, we investigated whether using constraints to break the symmetry these runs cause speeds up the solution of the OF, the ARF, and two formulations involving the OF and two kinds of symmetry breaking constraints for the blocks (i.e., the OF-VR-HO and OF-LEX(15)-HO approaches). The results are summarized in Table 10. The three arrays involve 4, 5 and 6 factors and 16, 32 and 44 distinct runs (out of 48), respectively. In some cases, adding the symmetry breaking constraints had a major impact on the computing times. For instance, the ARF was sped up substantially by adding the constraints for the 5-factor array when the goal was to obtain an arrangement in four or six blocks, and for the 6-factor array when looking for an arrangement in four blocks. Adding the symmetry breaking constraints for identical runs to the OF leads to an increase in computing time for all cases in which the computing time limit was not reached by the OF. In particular, for the design with six factors arranged in six blocks and the design with four factors arranged in twelve blocks, adding the constraints to the OF made that the corresponding problems could no longer be solved to optimality within the computing time limit of four hours, even though the plain OF could be solved to optimality well within four hours of

Table 8: Average computing times for finding optimal arrangements of 31 48-run designs in four and six blocks.

Formulation	4 blocks			6 blocks		
	Optimal solutions	CPU-T(s)	Gap	Optimal solutions	CPU-T(s)	Gap
OF	26/26	527.21	0.01%	13/13	3061.39	0.01%
	0/5	14400.12	0.56%	0/18	14401.65	1.19%
OF-LEX(14)	26/26	498.41	0.01%	13/13	2249.37	0.01%
	1/5	13418.81	0.52%	4/18	12630.48	0.79%
OF-LEX(15)	25/26	1020.69	0.01%	13/13	1814.69	0.01%
	1/5	13640.30	0.54%	2/18	13786.33	0.90%
ARF	23/26	2764.24	3.87%	13/13	4005.50	0.02%
	0/5	14400.48	3.08%	2/18	13478.50	4.21%
OF-LEX(15)-HO	26/26	747.87	0.00%	12/13	3658.84	3.68%
	1/5	13746.19	0.53%	2/18	13357.35	0.86%
OF-VR-HO	26/26	399.68	0.00%	13/13	1518.56	0.01%
	0/5	14400.08	0.45%	5/18	12535.79	0.72%
OF-OBJ	26/26	690.80	0.00%	11/13	3544.43	3.72%
	0/5	14401.09	0.57%	5/18	13403.32	0.81%

Table 9: Average computing times for finding optimal arrangements of 31 48-run designs in eight and twelve blocks.

Formulation	8 blocks			12 blocks		
	Optimal solutions	CPU-T(s)	Gap	Optimal solutions	CPU-T(s)	Gap
OF	9/9	885.76	0.00%	6/6	6976.54	0.01%
	0/22	14401.72	4.23%	0/25	14404.75	3.61%
OF-LEX(14)	9/9	256.85	0.00%	6/6	90.07	0.00%
	1/22	13839.65	3.67%	11/25	8590.34	0.90%
OF-LEX(15)	9/9	273.15	0.00%	6/6	198.67	0.00%
	1/22	13997.63	3.64%	11/25	8702.48	0.80%
ARF	9/9	54.30	0.00%	6/6	0.89	0.00%
	2/22	13367.36	3.41%	25/25	22.84	0.00%
OF-LEX(15)-HO	9/9	267.17	0.00%	6/6	159.90	0.00%
	1/22	14304.05	3.73%	11/25	8385.58	0.63%
OF-VR-HO	9/9	225.77	0.00%	6/6	68.51	0.00%
	1/22	13857.07	3.40%	13/25	8039.01	0.48%
OF-OBJ	9/9	352.06	0.00%	6/6	341.75	0.00%
	1/22	13830.03	3.86%	11/25	9234.57	0.84%

computing time. Combining constraints for breaking the symmetry due to the blocks and constraints for breaking the symmetry due to identical runs in the OF led to quite fast computing times for all designs with replicated runs arranged in four and six blocks. Overall, the combined OF-HO-VR-IR formulation worked best for the problems with four and six blocks, whereas the ARF-IR formulation worked best for problems with twelve blocks. None of the formulations allowed the solution of the instances with eight blocks within the computing time limit.

5.5 Additional experiments

5.5.1 Experimenting with the CPLEX symmetry breaking setting

In the previous sections, we have used the default CPLEX setting to conduct our experiments. As a consequence, the CPLEX symmetry breaking was also set at its default value. In order to study the impact of the various levels of symmetry breaking within CPLEX, we perform additional experiments: one in which the CPLEX symmetry breaking was turned off, and another one in which the CPLEX symmetry breaking was set to its most aggressive level. Using these two settings, we tested all formulations used in Tables 4 and 5 for the 32-run problems and all formulations used in Table 6 for the 40-run problems.

We observed that, for the OF, the computing times obtained by switching off the CPLEX symmetry breaking increase drastically (by a factor of 4 to 10), compared to using the default setting. To a lesser extent, this was also true for the OF-LEX(16) approach, which involves a weak symmetry breaking constraint. For all other formulations, the computing times are very similar using the two settings. The interpretation of these results is that the symmetry breaking that is used by default in CPLEX does not have added value if the problem formulation used already breaks the symmetry.

Another interesting result is that the impact of using formulations that break symmetry on the computing times is much larger than the impact of using the symmetry breaking options that CPLEX offers. For instance, CPLEX does not manage to solve the three difficult cases involving 32 runs and eight blocks within the computing time limit when the default symmetry breaking is utilized. However, switching off the symmetry breaking in CPLEX and using our formulations allowed the three difficult cases to be solved to optimality in almost all cases. This implies that, even though the default symmetry breaking in CPLEX is doing a very good job for the original formulation, substantial improvements in computing times can still be gained by adding the right symmetry breaking constraints or using a reformulation.

When comparing CPLEX's default symmetry breaking setting to its most aggressive symmetry breaking level, we observe that they generally result in very similar computing times. There are some exceptions, especially for the OF, where the aggressive setting even increases the computing time. Similar results were also found by Jans and Desrosiers (2013) for the job grouping problem. This leads us to the conclusion that the default symmetry breaking setting was an appropriate choice for our computational experiments.

5.5.2 Instances with permuted rows or columns

Jans and Desrosiers (2013) observed that, for the job grouping problem, the ordering of the input can have an effect on the computing times when symmetry breaking constraints are present. We tested the impact of the input ordering using the 32-run instances. We created three types of new instances by permuting the rows or the columns of the input matrix $\mathbf{X}$. We created the first type of permuted instances by randomly permuting the rows of $\mathbf{X}$, and the second type by randomly permuting the columns of $\mathbf{X}$. Finally, we created the third type of permuted instances using several systematic ways of ordering (such as increasing or decreasing orders according to a specific criterion). For all the newly generated instances, we performed the tests with the OF-LEX (14), OBJ and ARF formulations, as these formulations are good representatives for the different efficient symmetry breaking strategies in our paper.

From the experiments, we observe that all three types of permutations lead to a large variability in computing time. For the random row or column permutations, we observe that, for the OF-LEX(14) and OBJ formulations, our original instances provide a good performance, without being the best or worst. For the ARF, the original instances provided the best overall performance. Compared to the systematic orderings, our original ordering provides a medium performance for the OF-LEX(14) and OBJ formulations, whereas, for the ARF, our original ordering is very close to the best performance.

Table 10: Computing times for finding optimal blocking arrangements of 48-run orthogonal arrays with replicated runs.

Blocks	Factors	OF	OF-IR	OF-VR-HO	OF-VR-HO-IR	OF-LEX(15)-HO	OF-LEX(15)-HO-IR	ARF	ARF-IR
4	4	0.03	0.08	0.65	0.23	1.95	0.37	4.93	0.34
	5	408.64	556.28	1528.34	254.02	7309.96	356.20	14400.06	2351.42
	6	44.73	63.17	31.48	19.09	105.69	68.47	14400.05	9989.32
	Average	151.13	206.51	520.16	91.11	2472.53	141.68	9601.68	4113.69
6	4	0.11	0.12	1.64	0.40	1.98	0.30	0.58	0.50
	5	106.80	7424.32	1561.45	76.66	14400.03	558.87	10505.90	847.15
	6	3056.89	14400.51	3085.12	2323.15	10977.68	4202.37	14400.06	14400.08
	Average	1054.60	7274.99	1549.40	800.07	8459.90	1587.18	8302.18	5082.58
8	4	14401.79	14400.05	14400.05	14400.09	14401.08	14400.02	14400.03	14400.22
	5	14401.14	14400.09	14401.23	14400.05	14402.11	14400.05	14402.06	14402.20
	6	14401.84	14400.06	14400.05	14400.08	14402.25	14400.05	14400.23	14401.79
	Average	14401.59	14400.07	14400.44	14400.07	14401.81	14400.04	14400.77	14401.40
12	4	14.66	14400.11	11.23	5.46	28.61	8.60	0.33	7.63
	5	14401.19	14400.06	1545.17	622.35	661.37	662.30	1.03	5.99
	6	14402.45	14400.09	1027.80	750.01	328.33	2557.62	3.65	10.08
	Average	9606.10	14400.09	861.40	459.27	339.44	1076.17	1.67	7.90

ACCEPTED MANUSCRIPT

Given our results, it seems that there is no order that is consistently the best for all the various formulations. Therefore, it is certainly worth trying out different order rules for the instances.

5.5.3 Using GUROBI as a solver

We also experimented with GUROBI as a solver to see whether the results of our computational experiments would be drastically different. We used GUROBI version 7.0.2 with its default symmetry breaking. We did tests with all formulations presented in Tables 4 and 5 for the 32-run problems and all formulations from Table 6 for the 40-run problems.

We observed that for many problem instances, GUROBI has a better performance than CPLEX for the original formulation. This is, however, not a general trend as, for the 40-run problems with five blocks, CPLEX has the shortest CPU times. Across the various formulations we studied, GUROBI also exhibits much variability in its performance. Moreover, with GUROBI, our main conclusions still hold: we can generally improve the original formulation by adding specific symmetry breaking constraints, and we observe the same tendencies as for CPLEX. For example, for problems with a large number of blocks, we again observe that the ARF formulation is by far the best.

6 Conclusions

In this paper, we proposed various symmetry breaking constraints and an alternative formulation to tackle the symmetry in the original formulation of Sartono et al. (2015a) for identifying optimal arrangements of orthogonal arrays in blocks. We consider two kinds of symmetry: symmetry due to the blocks and symmetry due to the occurrence of identical runs in the orthogonal array under consideration.

We paid much attention to the first kind of symmetry, which is the most important one in screening experiments with many factors and limited numbers of runs. To ensure the symmetry due to the blocks is removed from the original problem formulation, we used variable reduction, lexicographical ordering and hierarchical ordering, and combinations of these techniques. We also studied the impact of objective perturbation, and we explored an asymmetric representatives formulation that does not suffer from symmetry due to the blocks. The asymmetric representatives formulation performed extremely well for large numbers of blocks and limited numbers of runs per block (e.g., eight, ten and twelve blocks of size four, and eight blocks of size six). However, that formulation performed rather poorly for problems involving few blocks (e.g., four blocks of size ten or twelve). If the number of blocks is as small as four, then the original formulation of Sartono et al. (2015a) is definitely competitive, and it cannot be improved upon by adding symmetry breaking constraints. Adding symmetry breaking constraints even tends to increase the required computing time, while the number of problems solved to optimality hardly increases. For larger numbers of blocks, adding symmetry breaking constraints to the original formulation does help to reduce the computing time. For instance, for problems with five blocks of eight runs, eight blocks of six runs, ten blocks of four runs or twelve blocks of four runs, all symmetry breaking constraints studied for the original formulation led to a decrease in computing time. The larger the number of blocks, the larger is the decrease in computing time. For the largest numbers of blocks, the best symmetry breaking approaches all allowed all problems to be solved to optimality well within the computing time limit.

An interesting result we obtained is that using the right kind of formulation to break symmetry due to the blocks has a substantially larger impact on the computing times than using the various symmetry breaking options embedded in the solver. As a result, there is much added value in the symmetry breaking approaches that we present in this paper. Another result we obtained is that using the most aggressive symmetry breaking option offered by the solver generally does not speed up the computations.

We also studied the use of symmetry breaking constraints to remove symmetry due to the occurrence of replicated runs in the orthogonal arrays. Adding these kinds of constraints to the asymmetric representatives formulation generally led to a substantial decrease in computing time except when arranging 48-run orthogonal arrays with replicated runs in 8 or 12 blocks. For the original formulation of Sartono et al. (2015a), adding constraints to remove symmetry due to identical runs led to an increase in computing time for almost all cases. However, the combination of constraints to break the symmetry due to the blocks and due to identical runs did lead to an improvement in computing time for designs with replicated runs arranged in four blocks of size ten, four blocks of size twelve, and six block of size eight. In conclusion, our results

concerning the use of symmetry breaking constraints to cope with identical runs are mixed. We therefore do not recommend to use these constraints by default.

Based on our computational experiments, we would recommend using the asymmetric representatives formulation whenever the number of blocks is large and the number of runs per block is at most six. For numbers of blocks up to four, there is no added value in using constraints that break the symmetry due to the blocks. For intermediate problems, the original formulation of Sartono et al. (2015a) combined with one or two types of symmetry breaking constraints seems best. The best symmetry breaking approaches involve the lexicographical ordering constraints in Equations (14) and (15), and variable reduction combined with hierarchical ordering.

In this paper, we illustrated our methodology by finding optimal arrangements of two-level orthogonal arrays with run sizes of 32, 40 and 48. In future research, it would be interesting to apply our recommended formulations to search for optimal blocking arrangements of mixed-level orthogonal arrays, and of two-level orthogonal arrays with run sizes larger than 48. Also, since we used the same objective function as Sartono et al. (2015a), our results are not optimal in terms of the statistical criteria called G- and G_2- aberration. This is because Sartono et al. (2015a) opted for an objective function that is linear in the decision variables. Finding an alternative objective function that is a better approximation to the G- and G_2- aberration criteria would be another interesting topic for future research. Finally, another interesting line of research is to develop a column generation approach using a formulation in which the variables are feasible blocks.

Acknowledgments

We gratefully acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (Grant 342182-09).

References

Adulyasak, Y., Cordeau, J. F., and Jans, R. (2014). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, 26:103–120.

Altinakar, S., Caporossi, G., and Hertz, A. (2016). A comparison of integer and constraint programming models for the deficiency problem. *Computers & Operations Research*, 68:89–96.

Bard, J. F. and Wan, L. (2008). Workforce design with movement restrictions between workstation groups. *Manufacturing & Service Operations Management*, 10:24–42.

Boland, N., Hughes, B. D., Merlot, L. T., and Stuckey, P. J. (2008). New integer linear programming approaches for course timetabling. *Computers & Operations Research*, 35:2209–2233.

Campêlo, M., Campos, V. A., and Corrêa, R. C. (2008). On the asymmetric representatives formulation for the vertex coloring polytope. *Discrete Applied Mathematics*, 156:1097–111.

Catanzaro, D., Godi, A., and Labbé, M. (2010). Class representative model for pure parsimony haplotyping. *INFORMS Journal on Computing*, 22:159–209.

Coelho, L. C. and Laporte, G. (2013). The exact solution of several classes of inventory-routing problems. *Computers & Operations Research*, 40:558 – 565.

Coelho, L. C. and Laporte, G. (2014). Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155:391–397.

Denton, B. T., Miller, A. J., Balasubramanian, H. J., and Huschka, T. R. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research*, 58:802–816.

Ghoniem, A. and Sherali, H. D. (2011). Defeating symmetry in combinatorial optimization via objective perturbations and hierarchical constraints. *IIE Transactions*, 43:575–588.

Hedayat, A. S., Sloane, N. J. A., and Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications*. Springer-Verlag, New York.

Ignizio, J. P. (1983). Generalized goal programming an overview. *Computers & Operations Research*, 10:227–289.

Jans, R. (2009). Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. *INFORMS Journal on Computing*, 21:123–136.

Jans, R., Degraeve, Z., and Schepens, L. (2008). Analysis of an industrial component commonality problem. *European Journal of Operational Research*, 186:801–811.

Jans, R. and Desrosiers, J. (2010). Binary clustering problems: symmetric, asymmetric and decomposition formulations. GERAD Technical Report G-2010-44, 15 pages.

Jans, R. and Desrosiers, J. (2013). Efficient symmetry breaking formulations for the job grouping problem. *Computers & Operations Research*, 40:1132–1142.

Junglas, D. (2007). Optimised grid-partitioning for block structured grids in parallel computation. PhD thesis, Darmstadt, Germany.

Kaibel, V., Peinhardt, M., and Pfetsch, M. E. (2011). Orbitopal fixing. *Discrete Optimization*, 8:595–610.

Kaibel, V. and Pfetsch, M. (2008). Packing and partitioning orbitopes. *Mathematical Programming*, 114:1–36.

Linderoth, J., Margot, F., and Thain, G. (2009). Improving bounds on the football pool problem by integer programming and high-throughput computing. *INFORMS Journal on Computing*, 21:445–457.

Margot, F. (2002). Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90.

Margot, F. (2003). Exploiting orbits in symmetric ILP. *Mathematical Programming*, 98:3–21.

Margot, F. (2010). Symmetry in integer linear programming. In Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., and Wolsey, L. A., editors, *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer Berlin Heidelberg.

Mee, R. W. (2009). *A comprehensive guide to factorial two-level experimentation*. Springer-Verlag, New York, NY, USA.

Melo, R. A. and Ribeiro, C. C. (2015). Improved solutions for the freight consolidation and containerization problem using aggregation and symmetry breaking. *Computers & Industrial Engineering*, 85:402–413.

Montgomery, D. C. (2009). *Design and analysis of experiments*. 7th edition, Wiley, New York.

Ostrowski, J., Anjos, M. F., and Vannelli, A. (2015). Modified orbital branching for structured symmetry with an application to unit commitment. *Mathematical Programming*, 150(1):99–129.

Ostrowski, J., Linderoth, J., Rossi, F., and Smriglio, S. (2011). Orbital branching. *Mathematical Programming*, 126:147–178.

Sartono, B., Goos, P., and Schoen, E. D. (2015a). Blocking orthogonal designs with mixed integer linear programming. *Technometrics*, 57:428–439.

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

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

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

Schoen, E. D., Sartono, B., and Goos, P. (2013). Optimum blocking for general resolution-3 designs. *Journal of Quality Technology*, 45:166–187.

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.

Schoen, E. D., Vo-Thanh, N., and Goos, P. (2018). Orthogonal blocking arrangements for 24-run and 28-run two-level designs. *Journal of Quality Technology*, 50:To appear.

Sherali, H. D., Fraticelli, B. M., and Meller, R. D. (2003). Enhanced model formulations for optimal facility layout. *Operations Research*, 51:629–644.

Van der Gaast, J. P., Rietveld, C. A., Gabor, A. F., and Zhang, Y. (2014). A tabu search algorithm for application placement in computer clustering. *Computers & Operations Research*, 50:38 – 46.

Wolsey, L. A. (1988). *Integer Programming*. Wiley.

Wu, C. F. J. and Hamada, M. S. (2009). *Experiments: Planning, Analysis, and Optimization*. 2nd edition, Wiley, New York, NY, USA.