Fachbereich II – Mathematik - Physik - Chemie

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

01/2018

Ulrike Grömping, Roberto Fontana

# An Algorithm for Generating Good Mixed Level Factorial Designs

Ein Algorithmus zur Erzeugung guter gemischt-stufiger faktorieller Versuchspläne (englischsprachig)

**Reports in Mathematics, Physics and Chemistry**

Berichte aus der Mathematik, Physik und Chemie

ISSN (print): 2190-3913

**Reports in Mathematics, Physics and Chemistry**

Berichte aus der Mathematik, Physik und Chemie

The reports are freely available via the Internet:
*http://www1.beuth-hochschule.de/FB_II/reports/welcome.htm*

01/2018, January 2018

An Algorithm for Generating Good Mixed Level Factorial Designs
Ein Algorithmus zur Erzeugung guter gemischt-stufiger faktorieller Versuchspläne
(englischsprachig)

*Editorial notice / Impressum*

*Published by / Herausgeber:*
Fachbereich II
Beuth Hochschule für Technik Berlin
Luxemburger Str. 10
D-13353 Berlin
Internet: http://public.beuth-hochschule.de/FB_II/
E-Mail: fbiireports@beuth-hochschule.de

Responsibility for the content rests with the author(s) of the reports.
Die inhaltliche Verantwortung liegt bei den Autor/inn/en der Berichte.

ISSN (print): 2190-3913

## Abstract

We propose an algorithm for the creation of mixed level arrays with generalized minimum aberration (GMA). GMA mixed level arrays are particularly useful for experiments involving qualitative factors: for these, the number of factor levels is often a consequence of subject matter requirements, while a priori assumptions on a statistical model are not made, apart from assuming lower order effects to be more important than higher order effects. Our algorithm creates GMA arrays using mixed integer optimization with conic quadratic constraints. Fully achieving GMA is feasible for small problems only; for larger problems, the optimization task is reduced to considering the confounding of low-order effects only. We provide lower bounds for the lowest-order confounding (given the number of experimental runs). Where one of these bounds is actually attainable, our algorithm is often fast in providing an array which attains it. Examples illustrate the scope and usefulness of our algorithm, which is implemented in an R package, using one of two commercial optimizers.

Keywords: Experimental design; Orthogonal arrays; Generalized minimum aberration; Mixed integer optimization.

## 1 Introduction

Factorial experiments are often run using orthogonal arrays. For example, engineers make ample use of the collection of arrays provided by the Japanese engineer Genichi Taguchi (see e.g. NIST Sematech 2016). Mixed level experiments, i.e. experiments for which not all factors have the same number of levels, are common in applications, especially if some factors are qualitative in nature. If a particular mixed level experiment is required, availability of a suitable array can be an issue. It is common to create a factorial design from a subset of the columns of a published array, and Grömping (in press) discussed ways to improve this type of usage by optimizing the choice of columns.

Of course, optimization of column selection from an existing array cannot be better than the creation of a tailor-made optimized array for the task at hand. For experiments with qualitative factors, one will often not have a particular model in mind, but will aim for the estimation of main effects, perhaps also of two-factor interactions, assuming that lower order effects are more important than higher order effects. For such a context, Fontana (2017) introduced an algorithm for the creation of an orthogonal array in a given number of runs that fulfills the quality criterion “generalized minimum aberration” (GMA, see Section 2.1); his proposal relied on the complex coding (Bailey 1982) and on mixed integer optimization. This paper modifies and improves Fontana’s proposal: since outer products of effect model matrices are invariant to the choice of normalized orthogonal coding (see Grömping 2018), we need not rely on the complex coding. Choice of an arbitrary real-valued normalized orthogonal coding (see Section 2.1) allows us to substantially reduce the size of the optimization problem. Furthermore, we choose a pragmatic approach which makes the algorithm feasible for larger problems: due to the nature of the optimization problem to be solved by mixed integer optimization, establishing overall GMA is extremely time consuming or even impossible for many problems of relevant sizes. Therefore, we primarily aim for optimizing aberrations with the goal of minimizing confounding of lowest-order effects (see Sections 2.1 and 3.3). For these, we provide two lower bounds that allow to confirm optimality without lengthy optimization iterations, in case a bound is attained. There still remain cases for which neither confirmation of an optimum nor further improvement is possible; for these, we consider it valuable to provide "improved" arrays, even if they are not necessarily optimal. All these – GMA designs, designs with optimized lowest-order confounding and "improved designs" – are what we call "good designs".

This paper derives the algorithm for obtaining good designs and describes its application to different design requests. We exemplify easier and harder examples. All calculations have been done with R package DoE.MIParray (Grömping 2017b), which offers functions based on two different commercial solvers for mixed integer optimization problems (Mosek and Gurobi, as documented in Mosek ApS 2017 and Gurobi Optimization, Inc. 2017). Both vendors provide free academic licenses and R packages that support access from within R. Mixed integer optimization is an area of active research; thus, what seems currently extremely challenging may become easily doable with future improvements to mixed integer algorithms. It is therefore possible that the range of applicability for the algorithm will broaden over time.

Section 2 introduces the necessary fundamentals and some basic results. Section 3 describes the algorithm, and Section 4 applies it to the test cases of Fontana (2017) (4.1) and to further interesting mixed level requests for orthogonal arrays (4.2) or supersaturated strength 1 arrays (4.3). The discussion highlights the merits of our proposal and points to needs for further research.

## 2 Basics

An array for accommodating $m$ factors in $n$ experimental runs can be written as an $n \times m$ array $d$ of symbols. The $j$th factor has $s_j$ levels, which are denoted as $1, \dots, s_j$, and the $n$ rows of the array $d$ contain the factor level combinations for $n$ experimental runs. An orthogonal array of strength $t$ in $n$ runs with $m_1$ factors at $s_1$ levels, …, $m_k$ factors at $s_k$ levels is denoted as $\mathrm{OA}(n, s_1^{m_1} \dots s_k^{m_k}, t)$; if we want to avoid requesting orthogonality between factors, we speak of balanced arrays and use the analogous notation $\mathrm{BA}(n, s_1^{m_1} \dots s_k^{m_k}, t)$. For all BA's, each factor has each of its levels the same number of times. OA's fulfill the stricter balance requirement that each pair of factors has each level combination the same number of times. In statistical applications, one often considers the resolution $R$ (denoted by a roman numeral), which is $t + 1$. An array is called "supersaturated", if there are fewer rows than needed for accomodating all main effects degrees of freedom. Supersaturated arrays can have at most strength 1, and we will restrict attention to arrays with at least strength 1.

We denote (column) vectors as bold face lower case letters, matrices as bold face capitals; $\mathbf{1}_n$ or $\mathbf{0}_n$ denote column vectors of $n$ ones or zeroes, respectively. Comparison for vectors (e.g. $\mathbf{r} \geq \mathbf{0}_N$) is to be understood element wise. $\otimes$ denotes the Kronecker product, the superscript $\top$ denotes the transpose.

## 2.1 GMA and counting vectors

For the considerations to follow, it is helpful to represent the $n \times m$ array $d$ via a counting vector $\mathbf{r}$. The idea is simple: a full factorial array in the $m$ factors would have $N = \prod_{j=1}^{m} s_j$ runs. We define $\mathcal{D}$ as this full factorial in lexicographic order (i.e. levels of the $j$th factor from 1 to $s_j$, leftmost factor changing levels most slowly). We can then represent an $n \times m$ array $d$ by the $N \times 1$ vector $\mathbf{r}$ that contains for each row of $\mathcal{D}$ the frequency with which it is contained as a row in $d$. This vector is called the counting vector. Of course, the sum of all elements of $\mathbf{r}$ is $n$.

The quality criterion "generalized minimum aberration" (GMA) was introduced by Xu and Wu (2001) and is based on the generalized word length pattern (GWLP), which measures for each effect order (0=intercept=overall mean, 1=main effects, 2=two-factor interactions, ...) the amount of confounding with the overall mean. The GWLP is calculated from model matrices in "normalized orthogonal coding":

**Definition 1** (normalized orthogonal coding)**.** Let $\mathbf{M} = (\mathbf{1}_N \vdots \mathbf{M}_1 \vdots \ldots \vdots \mathbf{M}_m)$ denote the $N \times N$ model matrix of the full model for the unreplicated full factorial design $\mathcal{D}$, with $\mathbf{M}_j$ denoting the $N \times df(j)$ matrix of interaction effect columns for all $j$-factor interactions. Then $\mathbf{M}$ is said to have normalized orthogonal coding, if

(i) $\mathbf{M}_0 = \mathbf{1}_N$,

(ii) all columns of $\mathbf{M}_1$ (main effects columns) have mean 0 and squared Euclidean norm $N$ and are orthogonal to each other,

(iii) and the $df(j)$ columns of $\mathbf{M}_j$, $j > 1$, are obtained as the products of $j$ columns from $\mathbf{M}_1$ that refer to $j$ distinct factors.

Note that Definition 1 implies that all columns of the matrix $\mathbf{M}$ have mean zero and squared norm $N$ and are orthogonal to each other. Furthermore, note that Definition 1 (iii) corresponds to the usual way of obtaining model matrix columns for interaction effects. The critical step is thus the choice of a coding for the main effects; examples of normalized orthogonal codings for 2-level, 3-level and 4-level main effects factors are given in Table 1. We observe that the complex coding obtained from the $s$th roots of the unity is another example of normalized orthogonal coding. For actual designs (called fractions $\mathcal{F}$ in the following), the full model model matrix $\mathbf{M}_{\mathcal{F}} = (\mathbf{1}_n \vdots \mathbf{M}_{\mathcal{F};1} \vdots \ldots \vdots \mathbf{M}_{\mathcal{F};m})$ is obtained from $\mathbf{M}$ by omitting runs not present in the

Table 1: Contrast matrices for $s = 2, 3, 4$

| $s = 2$ | $s = 3$ | | $s = 4$ | | |
|---|---|---|---|---|---|
| -1 | $-\sqrt{3/2}$ | $-\sqrt{1/2}$ | $-\sqrt{2}$ | $-\sqrt{2/3}$ | $-\sqrt{1/3}$ |
| 1 | $\sqrt{3/2}$ | $-\sqrt{1/2}$ | $\sqrt{2}$ | $-\sqrt{2/3}$ | $-\sqrt{1/3}$ |
| | 0 | $\sqrt{2}$ | 0 | $\sqrt{8/3}$ | $-\sqrt{1/3}$ |
| | | | 0 | 0 | $\sqrt{3}$ |

fraction (and duplicating runs occurring multiple times, if applicable). $\mathbf{M}_{\mathcal{F}}$ has normalized orthogonal coding if the $\mathbf{M}$ it has been taken from complies with Definition 1.

The GWLP is denoted as $(A_0, A_1, \ldots, A_m)$, with $A_0 = 1$ (confounding of the overall mean with itself). $A_j$ is called the number of (generalized) "words" of length $j$, and it is calculated as the sum of squared column averages over all $df(j)$ columns of $\mathbf{M}_{\mathcal{F};j}$, where $\mathbf{M}_{\mathcal{F}}$ has normalized orthogonal coding. The strength $t$ of an array is any positive integer for which $A_1 = \cdots = A_t = 0$; one usually identifies $t$ with the maximum possible strength; the resolution of an array is the integer $R$ such that $A_1 = \cdots = A_{R-1} = 0, A_R > 0$; we thus have $R = t + 1$, as was mentioned before. GMA consists in minimizing $A_1$, $A_2$, $A_3$, $\ldots$ in turn, i.e., one first maximizes the resolution, and then minimizes the number of shortest words $A_R$, followed by the number of second shortest words $A_{R+1}$, and so forth.

With our notation, we can write $A_j = \frac{1}{n^2}\mathbf{1}_n^\top \mathbf{M}_{\mathcal{F};j}\mathbf{M}_{\mathcal{F};j}^\top \mathbf{1}_n = \frac{1}{n^2}\mathbf{r}^\top \mathbf{M}_j \mathbf{M}_j^\top \mathbf{r}$. Thus, minimizing $A_j$ is equivalent to minimizing the quadratic form $\mathbf{r}^\top \mathbf{M}_j \mathbf{M}_j^\top \mathbf{r}$ w.r.t. the choice of a non-negative $N \times 1$ integer counting vector $\mathbf{r}$ with sum $n$. The matrix $\mathbf{H}_j = \mathbf{M}_j \mathbf{M}_j^\top$ has some useful properties which help to make the optimization feasible. The following sub section derives these.

## 2.2 Properties of $\mathbf{H}_j$

Before looking at $\mathbf{H}_j$ itself, we consider properties of normalized orthogonal contrast matrices. Let $\mathbf{C}_s$ denote the $s \times (s-1))$ main effect contrast matrix for an $s$-level factor in normalized orthogonal coding (the matrices in Table 1 are examples of $\mathbf{C}_2$, $\mathbf{C}_3$ and $\mathbf{C}_4$). Grömping (2018) showed that outer products of effect model matrices are invariant to the choice of normalized orthogonal coding. This applies in particular also to the matrices $\mathbf{C}_s$ themselves, which are effect model matrices in a simple one-factor model with $s$ runs.

The following lemma provides a formula for the coding invariant matrix $\mathbf{C}_s\mathbf{C}_s^\top$ which will be useful for proving that the matrices $\mathbf{H}_j$ consist of integer elements only.

**Lemma 1.** *Let $\mathbf{C}_s$ denote the contrast matrix for an $s$-level effect in normalized orthogonal coding. Then $\mathbf{C}_s\mathbf{C}_s^\top$ has diagonal elements $s-1$ and off-diagonal elements $-1$.*

*Proof.* The proof is by induction. Because of the coding invariance of the outer product, we can use any particular normalized orthogonal coding and will use the normalized Helmert coding for which the first few contrast matrices are given in Table 1. For 2-level factors, the assertion is trivial. Assume that the assertion holds for an $s$-level factor. The contrast matrix $\mathbf{C}_{s+1}$ can be written as

$$\mathbf{C}_{s+1} = \left(\begin{array}{c:c} \sqrt{\frac{s+1}{s}}\mathbf{C}_s & -\sqrt{\frac{1}{s}}\mathbf{1}_s \\ \hdashline \mathbf{0}_{s-1}^\top & \sqrt{s} \end{array}\right).$$

For the $(s+1) \times (s+1)$ outer product, this implies

$$\mathbf{C}_{s+1}\mathbf{C}_{s+1}^\top = \left(\begin{array}{c:c} \frac{s+1}{s}\mathbf{C}_s\mathbf{C}_s^\top + \frac{1}{s}\mathbf{1}_s\mathbf{1}_s^\top & -\mathbf{1}_s \\ \hdashline -\mathbf{1}_s^\top & s \end{array}\right).$$

It remains to be verified that the top left block is of the stated form: for the diagonal elements, $\frac{s+1}{s} \cdot (s-1) + \frac{1}{s} = s$, for the off-diagonal elements $\frac{s+1}{s} \cdot (-1) + \frac{1}{s} = -1$. Thus, the stated form for $\mathbf{C}_s$ indeed implies the stated form for $\mathbf{C}_{s+1}$, which finishes the proof. $\square$

As an aside, note that the explicit representation for the outer product of $\mathbf{C}_s$ implies that there is the single non-zero eigenvalue $s$ with multiplicity $s-1$ and one zero eigenvalue with corresponding eigenvector the normalized vector $\mathbf{1}_s$.

$\mathbf{H}_j = \mathbf{M}_j\mathbf{M}_j^\top$ is the sum of the outer products of the $df(j)$ individual columns of $\mathbf{M}_j$, but can also be written as a sum of outer product matrices over all $\binom{m}{j}$ $j$-factor interactions, i.e. as $\sum_{S\subseteq\{1,\dots,m\},|S|=j} \mathbf{X}_{\mathcal{I}(S)}\mathbf{X}_{\mathcal{I}(S)}^\top$, where $\mathbf{X}_{\mathcal{I}(S)}$ denotes the model matrix from the interaction among the factors in the set $S = \{i_1, \dots, i_j\}$ for the unreplicated full factorial array $\mathcal{D}$. According to Grömping (2018), the summands of this sum are invariant to the choice of normalized orthogonal coding, which of course implies coding invariance of the sum as well. Thus, we can simply choose an arbitrary normalized orthogonal coding, be it the complex coding chosen in Fontana (2017) or another one like e.g. normalized Helmert coding, available as contr.XuWu in R package DoE.base. The coding-invariant matrices $\mathbf{H}_j$ (calculable from that coding) have rank $df(j)$, where $df(j)$ denotes the number of degrees of freedom for all $j$-factor interactions and coincides with the number of columns of $\mathbf{M}_j$. According to Grömping (2018), $\mathbf{X}_{\mathcal{I}(S)}\mathbf{X}_{\mathcal{I}(S)}^\top = \mathbf{X}_{i_1}\mathbf{X}_{i_1}^\top \star \dots \star \mathbf{X}_{i_j}\mathbf{X}_{i_j}^\top$, where $\star$ denotes the Hadamard or Schur (= element wise) product. Consequently, investigation of outer products of main effects model matrices will provide insights into the nature of $\mathbf{H}_j$. Let $\mathbf{X}_1$ denote the main effect model matrix of the first factor in the full factorial array $\mathcal{D}$; because of lexicographic order, $\mathbf{X}_1 = \mathbf{C}_{s_1} \otimes \mathbf{1}_{s_2 \cdot \dots \cdot s_m}$, and $\mathbf{X}_1\mathbf{X}_1^\top = (\mathbf{C}_{s_1}\mathbf{C}_{s_1}^\top) \otimes (\mathbf{1}_{s_2 \cdot \dots \cdot s_m}\mathbf{1}_{s_2 \cdot \dots \cdot s_m}^\top)$. Thus, $\mathbf{X}_1\mathbf{X}_1^\top$ consists of elements $s_1 - 1$ and $-1$ only, and all its diagonal elements are $s_1 - 1$. Analogous reasoning applies to all main effects model matrices, i.e. the outer product of $\mathbf{X}_j$ contains elements $s_j - 1$ and $-1$ only, with all diagonal elements equal to $s_j - 1$ (more could be said, but will not be used now). Consequently, $\mathbf{X}_{\mathcal{I}(S)}\mathbf{X}_{\mathcal{I}(S)}^\top$ has elements that can be obtained as products of $-1$ entries and the dfs of the factors involved; in particular, its diagonal elements are $\prod_{i=1}^{j}(s_{i_j}-1)$. The overall matrix $\mathbf{H}_j$ is the sum of such matrices over all $S$ with $|S|=j$. As such, it trivially has integer entries only. Some properties of the matrix $\mathbf{H}_j$ are summarized in the following lemma. Part (ii) of the lemma holds, because the full factorial array, represented by $\mathbf{r}=\mathbf{1}_N$, has $A_j=0$ for all $j>0$.

**Lemma 2.** *The matrix $\mathbf{H}_j=\mathbf{M}_j\mathbf{M}_j^\top$ has the following properties:*

*(i) It is positive semidefinite with rank df(j).*

*(ii) Its elements sum to zero.*

*(iii) All its elements are integers.*

The algorithm will make use of the factorization $\mathbf{H}_j=\mathbf{M}_j\mathbf{M}_j^\top$ with the $N\times df(j)$ matrix $\mathbf{M}_j$. This factorization brings a substantial advantage versus the proposal by Fontana (2017), who used the Cholesky decomposition $\mathbf{H}_j=\mathbf{U}_j\mathbf{U}_j^\top$ with square matrices $\mathbf{U}_j$ instead: since the number of additional variables needed in conic quadratic constraints (see Section 3) depends on the number of columns of $\mathbf{M}_j$ or $\mathbf{U}_j$, respectively, our formulation requires fewer variables and thus brings larger problems into reach. Nevertheless, the size $N$ of the full factorial limits usability of the algorithm.

## 2.3 Lower bounds for $A_R$ and sum of the $A_j$

Grömping and Xu (2014) reported a lower bound for $A_R$ in $R$ factor arrays of resolution $R$ (their Theorem 5); for $m$ factor arrays with $m>R$, a lower bound can be derived by summing the lower bounds over all $R$ factor subsets of the $m$ factors. The algorithm discussed later in this paper can make use of a lower bound for $n^2A_R$; this is provided in the following two propositions.

**Proposition 1.** Consider an array $d$ in $n$ runs and $m$ factors with resolution $R$ (i.e., with strength $t=R-1$). Then,

$A_R(d)\cdot n^2\geq\sum_{S\subseteq\{1,\ldots,m\},|S|=R}(\prod_{i\in S}s_i-r_S)\cdot r_S$, with $r_S$ the rest when dividing $n$ by $\prod_{i\in S}s_i$.

For example, if an OA(18, $2^13^3$, 2) is sought, there are three 3-factor sets $S$ with one 2-level factor and two 3-level factors each ($r_S=0$) and one 3-factor set $S$ with three 3-level factors ($r_S=18$). With Proposition 1, $A_3\cdot 18^2\geq 3\cdot(18-0)\cdot 0+(27-18)\cdot 18=9\cdot 18$, which implies $A_3\geq 0.5$.

Following Grömping and Xu (2014), the bound of Proposition 1 is fulfilled, if and only if all $R$ factor sets have weak strength $R$, i.e. have resolution $R$ and do not have duplicate rows. This is most often the case for relatively small mixed level arrays, for which the algorithm is most useful. For symmetric $s$-level arrays, the bound is zero whenever $n$ is a multiple of $s^R$, because $r_S=0$ for such situations.

For balanced supersaturated arrays, which have resolution II, Ai, Fang and He (2007) and Liu and Lin (2009, their Lemma 2) published equivalent lower bounds for the quality criterion "expected $\chi^2$ value" or "sum of $\chi^2$ values" (related to each other by the factor $\binom{m}{2}$ for the number of pairs) that are closely related to $A_2$: according to Grömping (2017a), $n$ times the $A_2$ contribution of a pair of factors equals the $\chi^2$ contribution of the pair. Thus, these lower bounds can be easily adapted for $A_2$ (multipliers $2/(nm(m-1))$ or $1/n$, respectively), see the following proposition. In general, the lower bound from Ai, Fang and He (2007) or Liu and Lin (2009) is sharper for supersaturated arrays, while the bound from Proposition 1 is sharper for arrays that are not saturated; this is illustrated by Figure 1 for 12-run arrays with one 3-level factor, one 4-level factor and $x$ 2-level factors (for $x < 6$, negative calculation results for the bound from Proposition 2 have been replaced by the trivial lower bound 0).

**Proposition 2.** Consider an array $d$ in $n$ runs and $m$ factors with resolution 2 (i.e., with strength 1), with factor $i$ at $s_i$ levels, $i = 1, \ldots, m$. Then,

$$A_2(d) \cdot n^2 \geq \frac{n^2}{2(n-1)} \left( \left(\sum_{i=1}^m s_i\right)^2 - (n - 1 + 2m) \left(\sum_{i=1}^m s_i\right) + m(m + n - 1) \right).$$

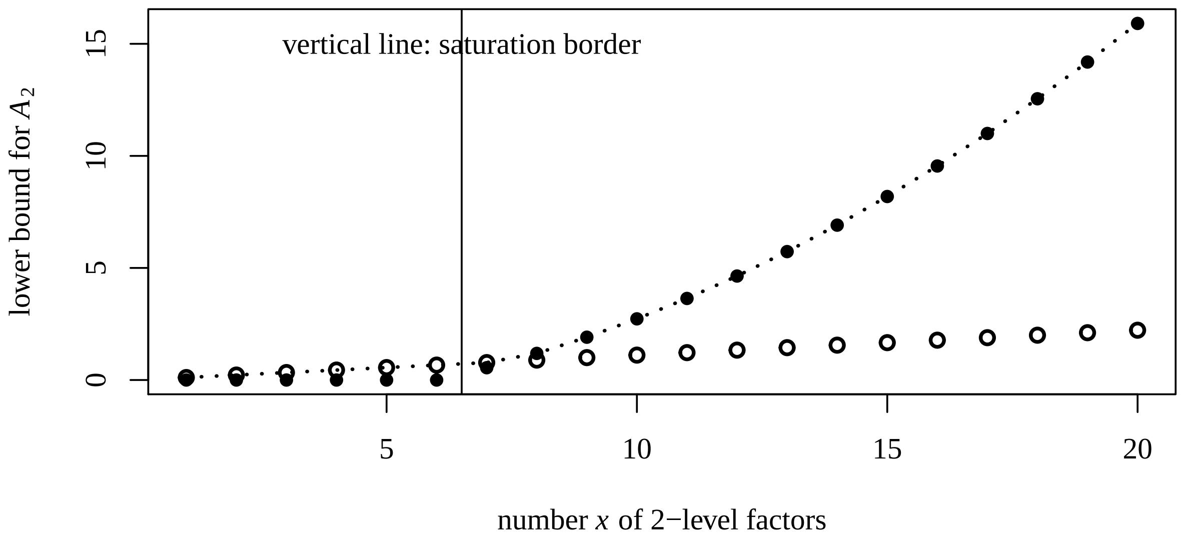

Figure 1: Bounds of Propositions 1 (open symbol) and 2 (closed symbol) for BA(12, $2^x 3^1 4^1$, 1)

Note that $n^2 A_2$ is always integral. Thus, in order for the bound of Proposition 2 to be sharp, it must be integral, which is often not the case; thus, it can be slightly sharpened for practical use.

The maximum of the bounds resulting from Propositions 1 and 2 can be used in the optimization process; since mixed integer optimization can be extremely slow to confirm optimality, its incorporation can be very helpful for situations for which it is in fact attained, because optimality is then known and does not need to be confirmed by reducing the gap to zero in the mixed integer algorithm (see Section 3).

In addition to the lower bounds for $A_R$, for arrays with distinct rows, i.e. counting vectors $\mathbf{r}$ with binary elements, a simple formula for the sum of the $A_j$ can be given:

**Proposition 3.** Consider $m$ factors with $s_1, \dots, s_m$ levels, for which $N = \prod_{j=1}^{m} s_j$ is the size of an unreplicated full factorial array $\mathcal{D}$. For any unreplicated $n$ run array $d$ in these $m$ factors, the sum of the GWLP entries is $\sum_{j=0}^{m} A_j(d) = N/n$.

In words, Proposition 3 states that the sum of the $A_j$ is the inverse proportion of the degrees of freedom needed for a full model (equal to the run size of the full factorial) that can be accomodated in an $n$ run array. For regular symmetrical $s$-level arrays, Proposition 3 follows from the number of generators needed, noting that each generator contributes $s-1$ to the sum (i.e. the number of words has to be multiplied with $s-1$ for obtaining the entries of the GWLP). For general symmetrical arrays, Proposition 3 directly follows from Xu and Wu's equation (8) with $j=0$, and noting that $B_0(d) = 1$ if and only if array $d$ does not have repeated rows and that $B_j'(d) = A_j(d)$. For mixed level arrays, the analogous reasoning is not explicated in Xu and Wu (2001), but does also apply (personal communication with Hongquan Xu). For obtaining GMA arrays, Proposition 3 implies that it suffices to optimize $A_1, \dots, A_{m-1}$, because $A_m = N/n - 1 - \sum_{i=1}^{m-1} A_i$, provided $d$ has distinct rows.

## 2.4 Quadratic cones

A cone is a set $C$ for which $a \in C \implies \forall \lambda \geq 0 : \lambda a \in C$. A quadratic cone is defined as $\mathcal{Q}^k = \{(x_1, x_2, \dots, x_k) : x_1 \geq \sqrt{x_2^2 + \cdots + x_k^2}\}$. A rotated quadratic cone is defined as $\mathcal{Q}_r^k = \{(x_1, x_2, \dots, x_k) : x_1, x_2 \geq 0 \text{ and } 2x_1x_2 \geq x_3^2 + \cdots + x_k^2\}$. For both $\mathcal{Q}^k$ and $\mathcal{Q}_r^k$, it is obvious that they are cones. $\mathcal{Q}_r^k$ has been written in the Mosek version of the definition (see Mosek ApS 2014), which is based on a proper rotation: the $(x_1, x_2)$ plane of $\mathcal{Q}^k$ is rotated such that the rotated coordinates $x_{1,r}$ and $x_{2,r}$ are proportional to $x_1 + x_2$ and $x_1 - x_2$, respectively, i.e. $x_{1,r} = (x_1 + x_2)/\sqrt{2}$ and $x_{2,r} = (x_1 - x_2)/\sqrt{2}$. With $x_{1,r}x_{2,r} = x_1^2/2 - x_2^2/2$, the relation between $\mathcal{Q}_r^k$ and $\mathcal{Q}^k$ is obvious. Gurobi omits the "2" on the left-hand side in $\mathcal{Q}_r^k$. This corresponds to the transformation $x_{1,r} = x_1 + x_2$ and $x_{2,r} = x_1 - x_2$ (not only rotated but also stretched by the factor $\sqrt{2}$), which implies $x_{1,r}x_{2,r} = x_1^2 - x_2^2$. This paper writes conic quadratic constraints in the Mosek notation; Gurobi's notation can be equivalently used by involving the multiplier 2 in appropriate places.

## 2.5 Quality criteria for mixed level arrays: ARFT and SCFT

Grömping (2017a) introduced refinements of $A_R$; these can be used for distinguishing non-isomorphic arrays with the same GWLP, and they are also usable as quality criteria, especially with a view to using an array as a screening design. Since these will be referenced in the Examples section, they are briefly reviewed here. Average $R^2$ frequency tables (ARFTs) and squared canonical correlation frequency tables (SCFTs) for resolution $R$ arrays are based on considering all $R$ factor sets.

Each $R$ factor set contributes an average $R^2$ value for each of its factors, and these values are tabulated in ARFT$_R$; the average $R^2$ value for an $s$-level factor in a set is the average over the $R^2$ values obtained from separate linear models for each of the $s-1$ main effects model matrix columns in an (arbitrary) orthogonal coding as responses, using the full model in the other $R-1$ factors from the set. Thus, an average $R^2$ value of "1" stands for complete confounding of the factor's main effect by the interaction of the other $R-1$ factors in the set. The larger the largest average $R^2$ value, the worse the worst-case confounding of a main effect by $R-1$ other factors (which is captured in the generalized resolution, see Grömping and Xu 2014). An array is considered better than another array, if its largest average $R^2$ value is smaller, in case of ties if the largest entry is less frequent, in case of ties, if the second largest average $R^2$ value is smaller, and so forth.

SCFTs go into more detail than ARFTs by considering individual degrees of freedom in a coding invariant way (worst-case coding): for each factor in an $R$ factor set, they tabulate the squared canonical correlations between the factor's main effects model matrix with the full model matrix of the other $R-1$ factors. Ones among the squared canonical correlations stand for the existence of a coding for which a main effect df is completely confounded by the full model in $R-1$ other factors. Comparison of arrays w.r.t. SCFTs proceeds in complete analogy to optimization w.r.t. ARFTs. Grömping (2017a) proposed to use ARFT$_R$ as the primary quality criterion and bring in SCFT$_R$ in case of ties.

Grömping and Bailey (2016) defined three new regularity types that are more lenient than conventional ones. These are related to SCFTs and ARFTs. In particular, SCFTs containing zeroes and ones only is a necessary criterion for all types of regularity, including the conventional ones. Grömping (2017a) argued that non-regular arrays are often preferrable for screening situations: if a resolution $R$ array is CC regular (i.e., has zero or one squared canonical correlations only), the one-valued squared canonical correlations imply that there is a main effect coding for which a particular main effect degree of freedom is completely confounded by an $R-1$ factor interaction. For arrays of resolution II, the presence of large average $R^2$ values or squared canonical correlations is particularly harmful, since it implies a strong relation between main effect estimates for two factors. Section 4.3 references a supersaturated array that was optimized elsewhere for weak equidistance and has a very beneficial ARFT$_2$ and SCFT$_2$ behavior. We will see various examples for which our algorithm yields arrays with good ARFTs and SCFTs.

# 3 The algorithm

## 3.1 The overall strategy

For achieving GMA, one has to successively request minimization of $A_1$, $A_2$, $A_3$, and so forth. Since each minimization may take a lot of effort, avoidance of minimization steps is a great time saver. We therefore implement a resolution option: if resolution $R$ is requested, the algorithm searches for a feasible counting vector that respects the requested resolution. This is a an integer-constrained linear optimization problem with constant objective function, which can usually be solved much faster than the conic quadratic optimization problems of later steps; the potential for this type of improvement was already mentioned in Fontana (2017). Furthermore, we allow the specification of a value $k_{max} \leq m-1$ (according to Proposition 3, $m$ is usually not needed) and limit optimization to $A_j$ with $j \leq k_{max}$, i.e. we restrict attention to a choosable maximum word length.

Thus, the algorithm has the following overall steps:

1. ensure the requested resolution $R_0$ (or establish that it is not feasible)
2. minimize $A_{R_0}$ (quadratic objective function), subject to the resolution constraints (linear constraints only), taking the solution of step 1 as the starting value. This step can make use of the lower bound that was introduced in Section 2.3: optimality can be ascertained if this bound is reached, which (if applicable) usually happens much faster than the confirmation of optimality by reducing the gap to zero (see below).
3. for $k_{max} > R_0$, minimize $A_j, R_0 < j \leq k_{max}$ (quadratic objective function), subject to the resolution constraints (linear) and all prior constraints obtained for $A_{j'}, R_0 \leq j' < j$ (quadratic inequality constraints).

Today's optimization software transforms minimization of a quadratic objective into minimization of a linear objective under a conic quadratic constraint, and quadratic constraints are likewise considered in this way. This is for example described in Mosek ApS (2014). The following section details the optimization problems to be solved for the algorithm outlined above.

General mixed integer optimization strategies are described in the documentations for Gurobi and Mosek (Gurobi Optimization, Inc. 2017 and Mosek ApS 2017); specific details are partly hidden to the public. In short, strategies combine solving a relaxed problem without integrality constraints and then trying to find integral solutions that are (almost) as good; heuristics are used in addition to simplex methods and branch and bound / branch and cut strategies, and optimality is proven, if the optimum of the relaxed problem equals the integral optimum. The discrepancy between relaxed and integral problem is called the "gap", and it is usually considered in relative terms as

$$\text{gap} = \frac{\text{current integral minimum} - \text{current relaxed minimum}}{\text{current integral minimum}}.$$

The current relaxed minimum is (of course) a lower bound for the integral minimum; providing the algorithm with a known more aggressive lower bound helps to prove optimality; with Gurobi and Mosek, the gap is nevertheless calculated based on the current relaxed minimum rather than a user-provided lower bound. Optimality is considered confirmed, when the gap becomes zero or a user-provided lower bound is reached. Thus, mixed integer optimization has two tasks: driving the current minimum as far down as possible and proving its optimality by reducing the gap. In many examples, reaching the actual optimum happens much faster than proving its optimality.

## 3.2 The sequence of optimization problems

The resolution step has to solve the optimization problem (1) for a specified resolution $R_0$. This linear problem with constant objective is often solved fast, or its infeasibility is established. In this and all further problems, it is also possible to constrain $\mathbf{r}$ to be binary instead of only integral, which guarantees an array with distinct rows. Once a feasible $\mathbf{r}$ for a resolution $R_0$ array has been found, the quadratic optimization problem of step 2 minimizes $A_{R_0}$, starting from the solution of problem (1). This corresponds to solving the quadratic optimization problem (2).

(1) $\min 0$

subject to

$\mathbf{1}_N^\top \mathbf{r} = n,$

$\mathbf{M}_1^\top \mathbf{r} = \mathbf{0}_{df(1)}, \ldots, \mathbf{M}_{R_0-1}^\top \mathbf{r} = \mathbf{0}_{df(R_0-1)},$

$\mathbf{r} \geq \mathbf{0}_N,$

$\mathbf{r}$ integral.

(2) $\min \mathbf{r}^\top \mathbf{H}_{R_0} \mathbf{r}$

subject to

$\mathbf{1}_N^\top \mathbf{r} = n,$

$\mathbf{M}_1^\top \mathbf{r} = \mathbf{0}_{df(1)}, \ldots, \mathbf{M}_{R_0-1}^\top \mathbf{r} = \mathbf{0}_{df(R_0-1)},$

$\mathbf{r} \geq \mathbf{0}_N,$

$\mathbf{r}$ integral.

Problem (2) can be rewritten as a linear problem with conic quadratic constraint using the following considerations:

- $\mathbf{H}_{R_0} = \mathbf{M}_{R_0}\mathbf{M}_{R_0}^\top$, with the $N \times df(R_0)$ matrix $\mathbf{M}_{R_0}$.
- Define $df(R_0)$ variables (not restricted to be non-negative) and collect them in the $df(R_0) \times 1$ vector $\mathbf{y}^{R_0}$. Restrict these by the equation $\mathbf{y}^{R_0} = \mathbf{M}_{R_0}^\top \mathbf{r} \iff \mathbf{M}_{R_0}^\top \mathbf{r} - \mathbf{y}^{R_0} = \mathbf{0}_{df(R_0)}$. This adds $df(R_0)$ additional variables and $df(R_0)$ additional equality constraints.

- Define two further non-negative variables $t_1^{R_0}$ and $t_2^{R_0}$, and request $(t_1^{R_0}, t_2^{R_0}, y_1^{R_0}, \dots, y_{df(R_0)}^{R_0})$ to be from the rotated cone $\mathcal{Q}_r^{df(R_0)+2} = \{(x_1, \dots, x_{df(R_0)+2}) : 2x_1x_2 \geq x_3^2 + \dots + x_{df(R_0)+2}^2, x_1, x_2 \geq 0\}$. Restrict $t_2^{R_0} = 0.5$. Then the cone constraint is equivalent to $t_1^{R_0} \geq \mathbf{y}^{R_0\top}\mathbf{y}^{R_0} = \mathbf{r}\mathbf{H}_{R_0}^\top\mathbf{r}$. Remember that the cone definitions used by different softwares are slightly different; for example, Mosek uses the above definition, Gurobi omits the "2". This difference can be equalized by restricting $t_2^{R_0}$ to 0.5 in Mosek and to 1 in Gurobi.
- As the objective, choose minimization of $t_1^{R_0}$; note that $t_1^{R_0}$ is guaranteed to be integral for integral $\mathbf{r}$, because $\mathbf{H}_{R_0}$ was shown to be integral.

$$
\begin{aligned}
(3)\quad & \min t_1^{R_0} \\
& \text{subject to} \\
& \mathbf{1}_N^\top \mathbf{r} = n, \\
& \mathbf{M}_1^\top \mathbf{r} = \mathbf{0}_{df(1)}, \\
& \dots, \\
& \mathbf{M}_{R_0-1}^\top \mathbf{r} = \mathbf{0}_{df(R_0-1)}, \\
& \mathbf{M}_{R_0}^\top \mathbf{r} - \mathbf{y}^{R_0} = \mathbf{0}_{df(R_0)}, \\
& \mathbf{r} \geq 0, \\
& t_1^{R_0} \geq 0, t_2^{R_0} = 0.5, \\
& (t_1^{R_0}, t_2^{R_0}, y_1^{R_0}, \dots, y_{df(R)}^{R_0})^\top \in \mathcal{Q}_r^{df(R)+2}, \\
& \mathbf{r} \text{ and } t_1^{R_0} \text{ integral.}
\end{aligned}
$$

$$
\begin{aligned}
(4)\quad & \min t_1^{R_0+1} \\
& \text{subject to} \\
& \mathbf{1}_N^\top \mathbf{r} = n, \\
& \mathbf{M}_1^\top \mathbf{r} = \mathbf{0}_{df(1)}, \\
& \dots, \\
& \mathbf{M}_{R_0-1}^\top \mathbf{r} = \mathbf{0}_{df(R_0-1)}, \\
& \mathbf{M}_{R_0}^\top \mathbf{r} - \mathbf{y}^{R_0} = \mathbf{0}_{df(R_0)}, \\
& \mathbf{M}_{R_0+1}^\top \mathbf{r} - \mathbf{y}^{R_0+1} = \mathbf{0}_{df(R_0+1)}, \\
& \mathbf{r} \geq 0, \\
& t_{1;min}^{R_0} + 0.3 \geq t_1^{R_0} \geq 0, t_2^{R_0} = 0.5, \\
& t_1^{R_0+1} \geq 0, t_2^{R_0+1} = 0.5, \\
& \left(t_1^{R_0}, t_2^{R_0}, y_1^{R_0}, \dots, y_{df(R_0)}^{R_0}\right)^\top \in \mathcal{Q}_r^{df(R_0)+2}, \\
& \left(t_1^{R_0+1}, t_2^{R_0+1}, y_1^{R_0+1}, \dots, y_{df(R_0+1)}^{R_0+1}\right)^\top \\
& \qquad \in \mathcal{Q}_r^{df(R_0+1)+2}, \\
& \mathbf{r}, t_1^{R_0} \text{ and } t_1^{R_0+1} \text{ integral.}
\end{aligned}
$$

The conic-quadratic version (Mosek variant) of the quadratic optimization problem (2) can thus be re-written as (3). Assuming that (3) has a solution with positive optimum and $k_{max} > R_0$, the solution vector to (3) can be used as a feasible starting value for the minimization of $A_{R_0+1}$, which is addressed by the optimization problem (4). The quadratic objective in (4) is treated analogously to the previous one, i.e. $df(R_0+1)+2$ additional variables are added, with $df(R_0+1)+1$ equality constraints and one conic quadratic constraint, and the first additional variable becomes the objective which is to be minimized. In addition, the previous constraints are retained, with the exception of the constraint on $t_1^{R_0}$, which must be adjusted: denoting as $t_{1;min}^{R_0}$ the optimized objective value of problem (3), one can request $t_1^{R_0} = t_{1;min}^{R_0}$; however, if the first step has been stopped for a timing constraint (not recommended, see Section 3.3), the constraint for variable $t_1^{R_0}$ should rather be chosen as $0 \leq t_1^{R_0} \leq t_{1;min}^{R_0}$, in order to allow improvements at later steps. Note that $t_{1;min}^{R_0}$ is an integer, so that an improvement will be noticeable, and a liberal tolerance can be added to the right-hand side without risking deterioration (we choose 0.3). Should $k_{max}$ be larger than $R_0 + 1$, problem (4) can be extended with further variables and constraints, analogously to the extension from (3) to (4).

Where $t_{1;min}^{R_0}$ turns out to be zero in problem (3) (i.e. better than requested resolution is feasible), the variables $t_1^{R_0}$, $t_2^{R_0}$ and $\mathbf{y}$ and all related constraints (including the conic quadratic one) can be removed, while adding the linear constraint $\mathbf{M}_{R_0}^\top \mathbf{r} = \mathbf{0}_{df(R_0)}$. In that case, the problem to be solved next is again of type (3), with increased $R_0$.

As was mentioned before, our approach representing $\mathbf{H}_j$ by $\mathbf{M}_j\mathbf{M}_j^\top$ in conification of the quadratic problem brings a substantial reduction in the number of variables used versus Fontana's (2017) Cholesky decomposition, which would require $N + 2$ additional variables per optimized $A_j$ in the optimization model. For full GMA optimization, we require $2(k_{max} - R_0 + 1) + \sum_{j=R_0}^{k_{max}} df(j) \leq 2(k_{max} - R_0 + 1) + N$ additional variables for optimizing $(A_{R_0}, \dots, A_{k_{max}})$ after an initial resolution step; this is much less than the $(k_{max} - R_0 + 1)(N + 2)$ additional variables that would be required when using a Cholesky decomposition (as in Fontana 2017), which does not exploit the rank defects of the $\mathbf{H}_j$.

## 3.3 Practical aspects

The algorithm is implemented in the R package DoE.MIParray; depending on the choice of $k_{max}$, it optimizes low order confounding only or (with $k_{max} = m - 1$ (or $m$)) can be used to ensure GMA. However, there are resource-driven practical problems. Running optimization without explicit time constraints will often run for a very long time, with an eventual interrupt by the user, risking to loose results. Running with a time constraint guarantees that the result obtained will be stored, but unfortunately hot-starting is of limited value: one can use the latest solution as a starting value, but it is not possible to preserve the branch and bound interim results, which will have to be re-established, when re-starting the algorithm.

In many applications, obtaining GMA for all word lengths will not be feasible. Generally, it should be considered more important to optimize shorter word lengths. For example, if $A_R$ can be optimized, optimization of $A_{R+1}$ is of interest; otherwise, one would rather further improve $A_R$. Whenever the lower bound according to Propositions 1 or 2 is reached (applicable for the shortest words only), one can safely switch to the next word length; for other cases, it is often not easy or even impossible to achieve the conviction that an optimum has been reached. It should usually be sufficient for practical purposes to optimize $A_R$ and $A_{R+1}$ (provided $R$ is the resolution), or $A_R$ only for the more difficult cases. If even optimality of $A_R$ cannot be reached or confirmed, an array constructed by the algorithm may still be useful; in such cases, it should be compared to arrays obtained from other approaches, e.g. the column selection optimization implemented in R package DoE.base (Grömping in press).

Clearly, it is desirable to have an array consist of distinct runs only. Thus, it is worth a try to request binary instead of integer elements for the vector **r**. Such a request, if feasible, should implicitly lead to better optima, because GMA arrays have unreplicated runs, where possible. Therefore, R package DoE.MIParray uses a restriction of **r** to binary as the default and allows relaxation to general integer as an option. Sometimes, the optimum array with distinct runs is nevertheless more easy to find by initially permitting repeat runs, i.e. general integer elements of **r**. Playing with optimizers' options can also sometimes help to solve difficult cases, particularly with Gurobi. And, although the optimization problem (of course) remains the same regardless of the order in which the variables are entered, optimization speed and the quality of the improvement reached by an optimizer within reasonable time does sometimes (strongly) depend on that order (see also the next section).

## 4 Examples

The calculations were done using two threads on a Windows machine with i7 processor with four cores and 32GB RAM, using the R package DoE.MIParray with R 3.4.1.

### 4.1 Test cases from Fontana (2017)

We start by addressing the test cases investigated in Fontana (2017). First, we consider accommodating five 2-level factors in 4 (not in Fontana 2017), 6, 8, 10, 12, 14 or 16 runs. We ran the algorithm from scratch without using any prior knowledge (i.e. optimizing $A_1$, $A_2$, and so forth, up to $A_5$, as proposed in Fontana 2017), or we requested each array to have the correct resolution $R$, so that only $A_R$ to $A_5$ needed to be optimized (we ignored the fact that optimization of $A_5$ can be skipped due to Proposition 3). For both these approaches, we applied the default settings in the R package DoE.MIParray, using both the optimizers Mosek and Gurobi (using package DoE.MIParray's functions gurobi_MIParray and mosek_MIParray, respectively). In all cases, the correct GMA array was found. The run times are reported in Table 2. The table shows that running from scratch can take substantially longer than starting from the correct resolution, particularly for the two resolution II cases in 10 and 14 runs with Mosek. It can also be seen that there is no general advantage of one software over the other. The run times we found for our implementation are substantially shorter than those reported in Fontana (2016); a portion of the improvement is due to a more powerful computer, another small portion may be due to the fact that Fontana accessed R via SAS or that the latest version of Mosek is faster, but the main contributors are most likely the smaller matrices and in particular

Table 2: Run times (s) for finding the GMA array for five 2-level factors

| $n$ | $R$ | $A_R$ | bound | Fontana 2017 | Mosek all | Gurobi all | Mosek from $A_R$ | Gurobi from $A_R$ |
|---|---|---|---|---|---|---|---|---|
| 4 | II | 2.000 | 1.688 | | 2.94 | 2.17 | 1.76 | 1.98 |
| 6 | II | 1.111 | 1.111 | 28 | 13.12 | 24.39 | 10.25 | 18.65 |
| 8 | III | 2.000 | 0.000 | 8 | 0.60 | 0.44 | 0.43 | 0.37 |
| 10 | II | 0.400 | 0.400 | 96 | 32.83 | 5.13 | 0.62 | 1.86 |
| 12 | III | 1.111 | 1.111 | 9 | 1.22 | 1.11 | 0.62 | 0.86 |
| 14 | II | 0.204 | 0.204 | 3415 | 126.86 | 18.94 | 1.28 | 3.65 |
| 16 | V | 1.000 | 1.000 | 29 | 0.35 | 0.28 | 0.15 | 0.12 |

the incorporation of the lower bound for confirming optimality. Fontana (2017) also reported a search time of 26 min for finding an OA(16, $2^6$, 3); the longest time used for this optimization with our implementation was below 2 s, even without exploiting knowledge about the resolution (but with exploiting lower bounds for confirming optimality).

It can be recommended to generally make an aggressive guess of the resolution, since infeasibility is often detected very fast. For all arrays of Table 2, the next larger resolution violates one of the known rules for the existence of arrays; as these rules are enforced before starting the optimization (using function oa_feasible from R package DoE.base), no time will be lost by lengthy searches for infeasible arrays. There are also other cases, however: for example, there is no OA(54, $3^6$, 3) (see e.g. the website by Eendebak and Schoen 2010), although none of the feasibility bounds is violated; the infeasibility of this array is not established fast by either Mosek or Gurobi.

Fontana (2017) created a GMA OA(18, $2^1 3^3$). Following the recommendation for an aggressive choice of resolution, an initial attempt for resolution IV was quickly rejected as infeasible. The GMA array was easily found by both optimizers, and both found the same array, since the GMA array is unique (see the website by Eendebak and Schoen 2010). The array attains the lower bound $A_3 = 0.5$, which was already calculated in Section 2.3. This benefits our algorithm, whose run times were about 6 s (Mosek) or 19 s (Gurobi), respectively, even when running from scratch; Fontana (2017) reported 270 s.

Fontana (2017) also produced a GMA OA(24, $2^2 3^1 4^1$, 2) and reported a run time of 37 min. Our algorithm produced a confirmed GMA OA(24, $2^2 3^1 4^1$, 2) in under 10 s, since (again) the array attains the lower bound $A_3 = 1/9$.

## 4.2 Further mixed level orthogonal arrays

Creation of an OA(18, $2^1 3^4$, 2) instead of the OA(18, $2^1 3^3$, 2) discussed in the previous section makes confirmed optimization a lot more difficult, because the GMA $A_3 = 3.5$ is larger than the lower bound 2 from Proposition 1 (the GMA array is known e.g. from Butler 2005). Nevertheless, both Mosek and Gurobi find the GMA array in less than 1 s; however, they have difficulties confirming optimality: Mosek needed about 464 min for confirming optimality of $A_3$, Gurobi still had a gap of more than 50% after about 2525 min (running with option MIPfocus set to 3 for improving the gap); subsequent optimization of $A_4$ has not been tried, so that we only know from external sources that the arrays that were found are indeed optimal. The website by Eendebak and Schoen (2010) lists two non-isomorphic GMA OA(18, $2^1 3^4$), and the two optimizers found different ones, as can e.g. be seen from comparing their ARFTs or SCFTs (not shown).

Coverage of mixed level arrays of larger strength in existing sources is a little patchy; for example, Eendebak and Schoen (2010) provided an OA(64, $2^2 4^1 8^1$, 3), but did not comment on the existence of arrays with more 2-level factors. For three and four 2-level factors, such arrays exist with $(A_4, A_5) = (3, 0)$ or $(A_4, A_5, A_6) = (6, 1, 0)$, respectively, and our algorithm finds non-regular arrays with these GWLPs after short optimization times (optimality of $A_4$ confirmed with the bound of Proposition 1). Regular arrays for these two scenarii can be found with the method published in Kobilinsky et al. (2017); this method does not optimize $A_4$ and finds arrays with $(A_4, A_5) = (3, 0)$ and $(A_4, A_5, A_6) = (7, 0, 0)$, respectively (with R package planor).

Larger arrays with strength 2 and far from saturating main effects df can also not be found easily in published tables; for example, an optimized OA(72, $2^4 3^2 4^1$, 2) is neither found on the Eendebak and Schoen (2010) website nor on the Gupta et al. (2011) website. Grömping (in press) reported on the creation of such an array by optimizing column selection from an OA(72, $2^{43} 3^8 4^1 6^1$, 2) taken from the Kuhfeld (2009) catalog of arrays; this array reached $(A_3, A_4) = (0.451, 3.247)$ and was used for a biotechnological experiment (see Vasilev et al. 2014). Our algorithm optimizes the initial feasible $A_3$ value fast; the quality of the objective reached in short time strongly depends on the order of entering the variables for this example. Using Mosek, entering the level numbers in the order 2,2,2,2,3,4,3 actually enabled the algorithm to reach the globally optimal $A_3 = 0.074$ (as confirmed by the lower bound of Proposition 1) within less than a minute; many other orderings of level numbers led to $A_3$ values that were at least slightly worse than that of the design used in Vasilev at al. (2014). Optimization of the $A_4$ value was attempted, but to no avail ($A_4 = 221/54$ was not reduced), and it is unknown to us whether the $A_4$ value is optimal or not.

The remaining three example arrays of this section are related to experimentation with optimizer options from the development phase of R package DoE.MIParray: an OA(128, $2^2 4^3$, 4), and OA(96, $2^2 3^1 4^2$, 3) and an OA(48, $2^2 3^1 4^2$, 2). The factors to be considered were MIQCPMethod (2-level), MIPFocus (2-level) and Heuristics (4-level) for Gurobi parameters, and scenario (initially 4-level, later 3-level; different level patterns) and runsize (4-level, tiny to large with resulting numbers depending on the level pattern) for the experimental situation. A resolution V array for the initial level pattern requires 128 runs. It was created using Gurobi and Mosek, and creation was very fast, because its $A_5$ equals the lower bound 1. The array obtained from Gurobi was slightly better than that obtained using Mosek in terms of its SCFT$_5$. An array like this could also have been taken from the Eendebak and Schoen website (2010), which lists ten non-isomorphic GMA arrays, or could have been produced by the algorithm of Kobilinsky et al. (2017) (regular array without optimizing $A_5$, but also yielding the optimum $A_5 = 1$ in this case).

After further consideration, it was decided to drop one of the scenarii, because prior experimentation suggested that it would likely add substantial run time with only little information. For the resulting experimental situation, with factors at 2, 2, 3, 4, 4 levels, a resolution V array smaller than the 192 run full factorial is not possible (which follows from the fact that the least common multiple of $2 \cdot 2 \cdot 4 \cdot 4 = 64$ and $2 \cdot 3 \cdot 4 \cdot 4 = 96$ is 192). Since 192 runs were considered too many, a resolution IV half fraction in 96 runs was considered. The optimized array turned out to have an $A_4$ value of 1/9; this was considered acceptable, since follow-up runs could be used if deemed necessary for resolving potential ambiguities. This array was easily obtained both by Gurobi and by Mosek. Again, the array created by Gurobi had a slightly better SCFT than the one created by Mosek; this array was therefore used for experimentation. An array like this is not available on the website by Eendebak and Schoen (2010), which does not cover any 96 run arrays with resolution III or IV.

Had one been prepared to contend with resolution III (not very sensible in the software development situation at hand), 48 runs would have done the job. That array is much harder to optimize than the 96 run array. Mosek found the GMA $A_3 = 2/9$ (equal to the bound of Proposition 1) in about 45 seconds. Optimizing $A_4$ did not improve the initial value 2.333, although a better $A_4$ exists. Playing with the ordering of factor levels, it turned out that the order 4,3,2,2,4 yielded a better array faster: after less than 20 seconds, $(A_3, A_4, A_5) = (2/9, 17/9, 8/9)$ was reached. This level ordering was also beneficial for the speed of Gurobi's optimization, which improved from about 190 seconds (and only when omitting the restriction to binary variables) to less than 10 seconds; however, for Gurobi the $A_4$ value obtained under that level ordering was worse (23/9 instead of 20/9, both worse than the 17/9 reached by Mosek under the beneficial level ordering); instead, the level ordering 4,3,2,4,2 led to the same GWLP that was reached by Mosek, however in about 6 min. As the website by Eendebak and Schoen (2010) does not cover the situation for this array, it is not known to us whether this GWLP is optimal or whether its $A_4$ can be improved. The arrays found by Mosek and Gurobi differ in their ARFTs (Mosek slightly better) and SCFTs; following the recommendations from Grömping (2017), the array produced by Mosek is therefore (slightly) preferable.

## 4.3 (Supersaturated) Resolution II arrays

Besides orthogonal arrays, our algorithm can also produce (balanced) resolution II arrays that are not as widely used and published as orthogonal arrays. Several of the examples of Table 2 are of that nature. Balanced *supersaturated* arrays are special cases of such resolution II arrays, and Proposition 2 provided a lower bound for their $A_2$. Some sources provide supersaturated arrays (SSDs) with many more main effects df than there are experimental runs; we refrain from discussing the chances and risks of using SSDs and refer readers to Georgiou (2014) for such a discussion. Our algorithm is not suitable for the creation of arrays whose full factorials would be extremely large; smaller supersaturated arrays can, however, be successfully optimized with the help of our algorithm, since optimization of the sum or expectation of $\chi^2$ (see also Section 2.3) is equivalent to optimization of $A_2$.

For example, Liu and Liu (2012) provided a weak equidistant BA(12, $2^9 3^3$, 1); they did not explicitly optimize $A_2$; however, the array's $A_2 = 2.8333333$ is close to the lower bound 2.7291667 for $A_2$. A different array with the same $A_2$ and the same ARFT$_2$ and SCFT$_2$ (and slightly better SCFT$_3$) was obtained with our algorithm using Mosek (the algorithm was run with a time limit of one hour; an approximately 45 min presolve step was followed by brief improvements). For a related array reported by Liu and Liu with an additional 6-level factor, our computer was not capable of accommodating the resource requirements by our algorithm; the same is true for most other arrays provided in Liu and Liu (2012).

There are many smaller mixed level situations, for which a specific supersaturated array for the requirement at hand is not readily available. If a saturated OA can be found, a subset of its runs and columns can sometimes be used (see Xu 2015). An SSD for mixed level situations can also be constructed from the website by Gupta et al., using the limited listing of balanced mixed level SSDs that they provide; however, the result will generally not be optimal itself. For example, Gupta et al. (2011) list a BA(12, $2^{11} 12^1$, 1), which can be used for creating a BA(12, $2^9 3^1 4^1$, 1). The lower bounds for the $A_2$ of a BA(12, $2^9 3^1 4^1$, 1) are 1 (Proposition 1) or 1.9097222 (Proposition 2, already sharpened), respectively. Creation by optimizing the selected 2-level columns from the array on the website and allocating a full factorial in a 3-level and a 4-level factor to the 12-level factor yields $A_3 = 3.333$; our algorithm created an array with $A_2 = 2$ reasonably fast (under 10 min with Mosek), i.e. was quite competitive for this small mixed level SSD, even though optimality of the array has not been proven.

## 5 Discussion

Mixed integer optimization can provide confirmed GMA arrays for small problems. For medium size problems, we have seen that it is often possible to obtain a confirmed optimum for the shortest word length $A_R$; this happens, whenever the GMA word length attains the lower bound provided in Proposition 1. It is then possible to further optimize the second shortest word length; however, for word lengths other than the shortest, there is generally no lower bound different from zero, so that it is usually necessary to confirm optimality by reducing the gap to zero, which is often infeasible, as was e.g. seen in the 72 or 48 run examples (Section 4.2), where it remains an open question whether or not $A_4$ is optimal.

Obtaining optimized arrays by mixed integer optimization is particularly interesting for mixed level experimentation, in case published arrays do not cover the experimentation needs. Our algorithm is most useful, if the next larger strength is almost but not quite in reach. For such situations, a best available array is often not catalogued, and it is not unlikely that the lower bound of Proposition 1 is attained by the GMA $A_R$.

Good arrays without optimality confirmation can also be obtained for small supersaturated situations.

Conic quadratic mixed integer optimization is notoriously difficult. Besides playing with optimizer options (especially for Gurobi), the examples showed that the order in which the level numbers are provided to the optimizer can have a strong influence on success or failure of the optimization, and it is certainly worthwhile to play with that order in case of difficult optimization tasks. Current state-of-the-art optimizers continue to be actively developed; the speed and success chances of our algorithm may benefit from these developments, e.g. with coming versions of Mosek or Gurobi.

The supersaturated example from Liu and Liu (2012) showed that weak equidistance coincided with a good $\mathrm{ARFT}_2$ and $\mathrm{SCFT}_2$. Thus, it might be of interest to optimize these for mixed level supersaturated arrays. Most arrays yielded by our algorithm showed very reasonable $\mathrm{ARFT}_R$ and $\mathrm{SCFT}_R$ behavior. It would be of interest to investigate whether the algorithm's ability of yielding such useful behavior can be systematized, over and above the current haphazard way of simply leaving the outcome to luck. This would be particularly useful for supersaturated designs, for which the worst-case confounding between two factors is particularly detrimental.

## Acknowledgments

Ulrike Grömping's work was partially supported by Deutsche Forschungsgemeinschaft Grant GR 3843/2. We appreciate input by Hongquan Xu regarding validity of Proposition 3 for the sum of the $A_i$ values in the mixed-level case.

## 6 References

Ai, M., Fang, K.-T. and He, S. (2007). $E(\chi^2)$-optimal mixed-level supersaturated designs. *Journal of Statistical Planning and Inference* **137**, 306-316.

Bailey, R.A.B. (1982). The decomposition of treatment degrees of freedom in quantitative factorial experiments. *Journal of the Royal Statistical Society* **B 44**, 63-70.

Butler, N.A. (2005). Generalised minimum aberration construction results for symmetrical orthogonal arrays. *Biometrika* **92**, 485-491.

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

Eendebak, P. T. and Schoen, E. D., (2010). Orthogonal Arrays Website. http://www.pietereendebak.nl/oapackage/series.html. Last accessed February 27, 2017.

Fontana, R. (2013). Algebraic generation of minimum size orthogonal fractional factorial designs: an approach based on integer linear programming. *Computational Statistics* **28**, 241-253.

Fontana, R. (2017). Generalized minimum aberration mixed-level orthogonal arrays: A general approach based on sequential integer quadratically constrained quadratic programming. *Communications in Statistics –Theory and Methods* **46**, 4275-4284.

Georgiou, S.D. (2014). Supersaturated designs: a review of their construction and analysis. *Journal of Statistical Planning and Inference* **144**, 92 - 109.

Grömping, U. (2017a). Frequency tables for the coding invariant quality assessment of factorial designs. *IISE Transactions* **49**, 505-517.

Grömping, U. (2017b). DoE.MIParray: Creation of Arrays by mixed integer optimization. R package version 0.10. In: R Core Team (2017).

Grömping, U. (2018). Coding Invariance in Factorial Linear Models and a New Tool for Assessing Combinatorial Equivalence of Factorial Designs. *Journal of Statistical Planning and Inference* **193**, 1-14.

Grömping, U. (in press). R Package **DoE.base** for Factorial Designs. *Journal of Statistical Software.* Preprint at http://www1.beuth-hochschule.de/FB_II/reports/Report-2016-001.pdf.

Grömping, U. and Bailey, R.A. (2016). Regular fractions of factorial arrays. In: Kunert, J., Müller, C.H., Atkinson, A.C. (Eds.), *MODA11—Advances in Model-Oriented Design and Analysis.* Springer, pp. 143–151.

Grömping, U. and Xu, H. (2014). Generalized resolution for orthogonal arrays. *The Annals of Statistics* **42**, 918-939.

Gupta, V.K., Parsad, R., Kole, B. and Bhar, L.M. (2011). Supersaturated Designs. http://www.iasri.res.in/design/Supersaturated_Design/SSD/Supersaturated.html. Indian Agricultural Statistics Research Institute (ICAR), New Delhi 110 012, India. Last accessed September 22, 2017.

Gurobi Optimization, Inc. (2017). Gurobi Optimizer Reference Manual. http://www.gurobi.com/documentation/current/refman.pdf.

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

Kobilinsky, A., Monod, H. and Bailey, R.A. (2017). Automatic generation of generalised regular factorial designs. *Computational Statistics and Data Analysis* **113**, 311-329.

Kuhfeld, W. (2009). Orthogonal Arrays. Website courtesy of SAS Institute http://support.sas.com/techsup/technote/ts723.html. Last accessed September 25 2017.

Liu, M.Q. and Lin, D.K.J. (2009). Construction of Optimal Mixed-Level Supersaturated Designs. *Statistica Sinica* **19**, 197-211.

Liu, Y. and Liu, M.Q. (2012). Construction of equidistant and weak equidistant supersaturated designs. *Metrika* **75**, 33–53. DOI 10.1007/s00184-010-0313-9

Mosek ApS (2014). MOSEK Modeling Manual. http://docs.mosek.com/generic/modeling-a4.pdf.

Mosek ApS (2017). MOSEK Rmosek Package 8.1.0.24. http://docs.mosek.com/8.1/rmosek/index.html.

NIST/SEMATECH (2016). *e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/ handbook/, 21 August 2016.

R Core Team (2017). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

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. doi:10.1002/jcd.20236.

Vasilev, N., Schmitz, Ch., Grömping, U., Fischer, R. and Schillberg, S. (2014). Assessment of Cultivation Factors that Affect Biomass and Geraniol Production in Transgenic Tobacco Cell Suspension Cultures. *PLoS ONE* **9**(8): e104620. DOI:10.1371/journal.pone.0104620.

Xu, H., Cheng, S.-W. and Wu, C.F.J. (2004). Optimal projective three-level designs for factor screening and interaction detection. *Technometrics* **46**, 280–292.

Xu, H. (2005). A Catalogue of Three-Level Regular Fractional Factorial Designs. *Metrika* **62**, 259-281.

Xu, H. (2015). Nonregular factorial and supersaturated designs. In *Handbook of Design and Analysis of Experiments*, eds A. Dean, M. Morris, J. Stufken and D. Bingham, Chapman and Hall/CRC, Series: Chapman & Hall/CRC Handbooks of Modern Statistical Methods, 339-370.

Xu, H. and Wu, C.F.J. (2001). Generalized minimum aberration for asymmetrical fractional factorial designs [corrected republication of MR1863969]. *The Annals of Statistics* **29**, 1066–1077.