

# Generalized Minimum Abberation Designs with Small Run Sizes

Dursun A. Bulutoglu

Air Force Institute of Technology  
Wright-Patterson AFB, Ohio 45433  
dursun.bulutoglu@afit.edu

Kenneth J. Ryan

Bowling Green State University  
Bowling Green, Ohio 43403-0267  
kjryan@bgsu.edu

June 25, 2009

## ABSTRACT

Enumeration algorithms based on finding all non-isomorphic solutions to a sequence of integer linear programs (ILPs) are used to find generalized minimum aberration (GMA) designs by all classifying non-isomorphic orthogonal arrays:  $OA(24,k,2,2)$ ,  $OA(32,k,2,3)$ ,  $OA(40,k,2,3)$ ,  $OA(48,k,2,3)$ ,  $OA(81,k,3,3)$ , and  $OA(160,k,2,4)$ . In addition, a new directed enumeration technique is used to find GMA OAs by classifying good non-isomorphic  $OA(28,k,2,2)$  with  $k = 3-14$  and  $OA(36,k,2,2)$  with  $k = 3-9$  which are no worse than a pre-specified generalized word length pattern (GWP) under the GMA criterion. Properties of a GMA design for each  $OA(N, k, s, t)$  mentioned above are tabled. New, fast search algorithms which output GMA or near GMA OAs are tested against the new catalog of GMA designs and used to produce weak GMA  $OA(36, k, 2, 2)$  with  $k = 10-18$  and near GMA  $OA(28, k, 2, 2)$  with  $k = 15-26$ .

KEY WORDS: Isomorphism rejection; Nauty.

## 1. INTRODUCTION

A fractional factorial design  $\mathbf{Y}$  with  $N$  runs and  $k$  factors each having  $s$ -levels is an orthogonal array of strength  $t$ ,  $1 \leq t \leq k$ , denoted by  $OA(N, k, s, t)$ , if each of the  $s^t$  level combinations appears exactly  $N/s^t$  times when  $\mathbf{Y}$  is projected onto any  $t$  factors. The index  $\lambda$  of an  $OA(N, k, s, t)$  is defined as  $N/s^t$ . An  $OA(N, k, s, t)$  is universally optimal for estimating the model containing all main effects and all interactions having  $\lfloor t/2 \rfloor$  factors or less; see Cheng (1980) and Mukerjee (1982). When  $s$  is a prime power, regular fractional factorial designs (also known as linear codes), constructed by solving a linear system of equations over the field  $GF(s)$ , are examples of orthogonal arrays. Since a regular fractional factorial design is the solution set of a linear system of equations over  $GF(s)$ , the number of runs it contains must be a power of  $s$ . On the other hand, such a restriction does not apply to all orthogonal arrays. Even though there is a vast literature on the construction and existence of regular fractional factorial designs (see Hedayat, Sloane, and Stufken, 1999), not as much seems to be known about orthogonal arrays.

The design obtained by permuting factors or runs as well as levels in a subset of factors in an  $OA(N, k, s, t)$  is also an  $OA(N, k, s, t)$ . Two  $OA(N, k, s, t)$  are called *isomorphic* if one can be obtained from the other by applying a sequence of these operations. Assuming the hierarchical ordering principle (see Section 3.5 of Wu and Hamada, 2000), two  $OA(N, k, s, t)$  are compared under model uncertainty using the generalized minimum aberration (GMA) criterion developed in Xu and Wu (2001). Let  $\mathbf{Y} = [y_{ij}]$  be a 2-level design with entries  $\pm 1$  having  $N$  runs and  $k$  factors, and  $l = \{i_1, i_2, \dots, i_r\} \subseteq \mathbb{Z}_k := \{1, \dots, k\}$

be a nonempty subset of  $r$  factors. The GMA criterion is based on the concept of the  $J$ -characteristics

$$J_r(l) := \sum_{i=1}^N \prod_{j \in l} y_{ij}$$

introduced by Tang and Deng (1999). Note that  $0 \leq |J_r(l)| \leq N$  and that larger values  $|J_r(l)|$  imply a stronger degree of aliasing among the factors in  $l$ . An average aliasing among all subsets of  $r$  factors is

$$A_r(\mathbf{Y}) := \frac{1}{N^2} \sum_{\{l \subseteq \mathbb{Z}_k : |l|=r\}} J_r(l)^2,$$

and the sequence  $\text{GWP}(\mathbf{Y}) := (A_1(\mathbf{Y}), A_2(\mathbf{Y}), \dots, A_k(\mathbf{Y}))$  is called the generalized word length pattern (GWP) of  $\mathbf{Y}$ . The GMA criterion selects designs that sequentially minimize the GWP.

The general concept of GWP for  $s$ -level designs is computed as follows. Let  $d_{ij}(\mathbf{Y})$  be the number of columns at which the  $i$ th and  $j$ th rows of  $\mathbf{Y}$  differ, and define

$$B_r(\mathbf{Y}) := N^{-1} |\{(i, j) : d_{ij}(\mathbf{Y}) = r, i, j = 1, \dots, N\}|$$

for  $r = 0, \dots, k$ . The distance distribution  $(B_0(\mathbf{Y}), B_1(\mathbf{Y}), \dots, B_k(\mathbf{Y}))$  of  $\mathbf{Y}$  determines the components of GWP and vice versa; the direct relationships provided in Xu and Wu (2001) are:

$$\begin{aligned} A_j(\mathbf{Y}) &= N^{-1} \sum_{i=0}^k P_j(i, s, k) B_i(\mathbf{Y}) \\ B_j(\mathbf{Y}) &= N s^{-k} \sum_{i=0}^k P_j(i, s, k) A_i(\mathbf{Y}) \end{aligned} \quad (1)$$

for  $j = 0, \dots, k$ , where  $A_0(\mathbf{Y}) = 1$  and  $P_j(x, s, k) := \sum_{i=0}^j (-1)^i (s-1)^{j-i} \binom{x}{i} \binom{k-x}{j-i}$  are the *Krawtchouk polynomials*. When computing the Krawtchouk polynomials, the recursion

$$P_j(x, s, k) = P_j(x-1, s, k) - P_{j-1}(x-1, s, k) - (s-1)P_{j-1}(x, s, k)$$

with initial values  $P_0(x, s, k) = 1$  and  $P_j(0, s, k) = (s-1)^j \binom{k}{j}$  is convenient.

In general, finding GMA designs is a very difficult problem. Butler (2003, 2004) theoretically constructed 2-level GMA designs. Butler's proofs for establishing that the constructed designs are GMA involved finding lower bounds for the GWP of 2-level designs for certain number of runs and factors. Xu (2005) derived lower bounds for the GWP using linear programming in infinite precision and also found

2-level factorial designs based on the Nordstorm–Robinson code that achieve the bounds. Bulutoglu and Kaziska (2009) improved the lower bounds of Xu (2005) using ILP (Integer Linear Programming) in infinite precision developed by Espinoza (2006) and Applegate *et al.* (2007).

Another way of finding GMA designs is by classifying all non-isomorphic orthogonal arrays. If two  $\text{OA}(N, k, s, t)$  are isomorphic, they are indistinguishable under the GMA criterion. On the other hand, there are non-isomorphic  $\text{OA}(N, k, s, t)$  with the same GWP. Classifying all non-isomorphic  $\text{OA}(N, k, s, t)$  allows the best to be found with respect to the GMA or any other ordering criterion that remains invariant between isomorphic designs. Even though there have been classifications of non-isomorphic  $\text{OA}(N, k, s, t)$  (e.g., Seiden and Zemach, 1966; Hedayat, Seiden, and Stufken, 1997; Hedayat, Stufken, and Su, 1997; and many others) efficient algorithmic tools to push the frontier are lacking.

Bulutoglu and Margot (2008) showed that finding all  $\text{OA}(N, k, s, t)$  is equivalent to finding all non-negative integer solutions to a symmetric ILP with binary coefficients. Two solutions were defined to be *isomorphic* if they correspond to isomorphic orthogonal arrays, so finding all non-isomorphic solutions is equivalent to finding all non-isomorphic  $\text{OA}(N, k, s, t)$ . Branch-and-cut algorithms are a standard technique for solving ILPs (see Padberg and Rinaldi, 1991), but a presence of symmetry in an ILP may require extending the basic branch-and-cut algorithm to avoid solving isomorphic subproblems more than once. Such an extension was proposed by Margot (2002, 2003a, 2003b, 2007) and used by Bulutoglu and Margot (2008) to find all isomorphism classes of  $\text{OA}(24, 7, 2, 2)$ ,  $\text{OA}(24, k, 2, 3)$  for  $6 \leq k \leq 11$ ,  $\text{OA}(32, k, 2, 3)$  for  $6 \leq k \leq 11$ ,  $\text{OA}(40, k, 2, 3)$  for  $6 \leq k \leq 10$ ,  $\text{OA}(48, k, 2, 3)$  for  $6 \leq k \leq 8$ ,  $\text{OA}(56, k, 2, 3)$  for  $k = 6, 7$ ,  $\text{OA}(64, k, 2, 4)$  for  $k = 7, 8$ ,  $\text{OA}(80, 6, 2, 4)$ ,  $\text{OA}(96, 7, 2, 4)$ ,  $\text{OA}(112, 6, 2, 4)$ , and  $\text{OA}(144, 8, 2, 4)$ .

If an  $\text{OA}(N, k_0, s, t)$  does not exist, then no  $\text{OA}(N, k, s, t)$  exists for  $k \geq k_0$ . Hence, there is a maximum  $k$ , say  $f(N, s, t)$ , for which an  $\text{OA}(N, k, s, t)$  exists. Bulutoglu and Margot (2008) showed that  $f(80, 2, 4) = 6$ ,  $f(96, 2, 4) = 7$ ,  $f(112, 2, 4) = 6$ , and  $f(144, 2, 4) = 8$  by determining the smallest  $k$  for which the number of non-isomorphic  $\text{OA}(N, k, s, t)$  is zero. However, the number of variables in the ILP formulation of Bulutoglu and Margot (2008) is  $s^k$  which grows exponentially with  $k$ . This is a major drawback as the problem of finding at least one  $\text{OA}(N, k, s, t)$  becomes impractical for modestly large  $k$ .

We introduce four algorithms in Section 2 to circumvent the drawback mentioned in the last paragraph and find all non-isomorphic  $\text{OA}(N, k, s, t)$  for many  $(N, k, s, t)$  by sequentially enumerating non-isomorphic  $\text{OA}(N, k, s, t)$  for  $k = t, t + 1, \dots, f(N, s, t)$ . Our methods require finding all non-isomorphic solutions to sequences of symmetric ILPs. Sequences of symmetric linear systems of equations in non-negative integer variables (which can be viewed as sequences of ILPs) were also solved in Sun, Li, and Ye (2004), Angelopoulos *et al.* (2007), Evangelaras *et al.* (2007), and Angelopoulos *et al.* (2009). But these

prior methods i) are based on brute force enumeration and isomorphism rejection and do not take advantage of the dual simplex algorithm, isomorphism pruning, and zero setting in Margot (2007) and ii) are only for 2-level factorial designs. A new directed enumeration algorithm which captures all  $OA(N, k, 2, t)$  that are as good or better than a pre-specified GWP is defined in Section 3 to put more 2-level GMA designs within computational reach. Appendix A contains proofs of the methods in Sections 2 and 3.

A catalog of GMA OAs (obtained via the methods of Sections 2 and 3) is documented in Section 4. The GMA designs are available (<http://www.afit.edu/en/enc/Faculty/Dbulutoglu/Bulutoglu.html>) for practical use. For example, a minimum aberration (MA)  $2^{9-3}$  regular design is an  $OA(64, 9, 2, 3)$ , so a practitioner could avoid the sample size restrictions of regular designs and use the GMA  $OA(40, 9, 2, 3)$  from our catalog for a strength 3 design in fewer runs if a 40 run experiment exhausts the budget.

In Section 5, new heuristic search algorithms which quickly output GMA or near GMA  $OA(N, k, s, t)$  are validated against the Section 4 GMA design catalog. These heuristics were used to output weak GMA  $OA(36, k, 2, 2)$  and GMA or near GMA  $OA(28, k, 2, 2)$  for large  $k$ , when enumerations would take too long; these designs are also on the first author's webpage. Computations were done at the Ohio Supercomputer Center (OSC) with 2GHz processors and on the second author's computer with 3GHz processors and a 4TB hard drive. Assume an OSC processor with reported job times unless told otherwise.

## 2. EXTENSION ALGORITHMS FOR ENUMERATING $OA(N, k, s, t)$

For  $k = t, t+1, \dots, f(N, s, t)$ , let  $n_k$  be the number of non-isomorphic  $OA(N, k, s, t)$  and  $T_k = \{\mathbf{Y}_1, \dots, \mathbf{Y}_{n_k}\}$  be a set of all non-isomorphic  $OA(N, k, s, t)$ . The basic extension algorithm obtains  $T_k$  from  $T_{k-1}$ .

**Algorithm 2.1 (Basic Extension)** *Input:*  $N, k, s, t$ , and  $T_{k-1}$ . *Set*  $l := 1$ .

1. Obtain a set  $M_l$  of  $OA(N, k, s, t)$  such that the first  $k-1$  columns are the same as  $\mathbf{Y}_l \in T_{k-1}$  and at least 1 representative from each isomorphism class of such  $OA(N, k, s, t)$  is included.
2. Increment  $l := l + 1$  and then repeat Step 1 if  $l \leq n_{k-1}$ .
3. Set  $M := \bigcup_{l=1}^{n_{k-1}} M_l$ . Form  $T_k$  from  $M$  by picking one representative from each isomorphism class of  $OA(N, k, s, t)$  in  $M$ . *Output:*  $T_k$  and  $l = n_{k-1}$ .

The  $s^t$  full factorial design replicated  $\lambda := N/s^t$  times is the singleton  $T_t$ . Given  $T_{k-1}$ ,  $T_{k+m}$  is obtained after applying the basic algorithm  $m+1$  times. Alternatively, if  $k > t+1$ , Bulutoglu and Margot (2007) can be used to directly obtain input  $T_{k-1}$  of the basic algorithm. For Step 3, we applied the graph based

technique with the program nauty McKay (2007) described in Ryan and Bulutoglu (2009) due to its efficiency. The contribution here is to develop efficient versions of Step 1 for the basic algorithm.

Hereinafter, assume the factor levels of an  $OA(N, k, s, t)$  are coded  $0, \dots, s - 1$ . Sun *et al.* (2004) classified all  $OA(20, k, 2, 2)$  with a version of the basic algorithm which used an isomorphism rejection technique to reduce “over-representation” in Step 1. The necessary and sufficient condition for a column vector  $\mathbf{x} \in \{0, 1\}^{20}$  to extend an  $OA(20, k - 1, 2, 2)$  to an  $OA(20, k, 2, 2)$  is that the entries of  $\mathbf{x}$  sum to 10 and that the dot product of  $\mathbf{x}$  with each of the columns of the  $OA(20, k - 1, 2, 2)$  is 5. Hence, the design obtained by extending the  $OA(20, k - 1, 2, 2)$  with  $\mathbf{x}$  is an  $OA(20, k, 2, 2)$  if and only if  $\mathbf{x}$  satisfies a linear system of equations. Sun *et al.* (2004) checked each of the  $\binom{20}{10}$  possible vectors in  $\{0, 1\}^{20}$  that sum to 10 to see which satisfy these equations, but this is expensive and not necessary. Using the branch-and-cut enumeration technique is much less expensive.

The following lemma and definition are used to formulate the ILP feasibility problem resulting from extending an  $OA(N, k - 1, s, t)$  to an  $OA(N, k, s, t)$ .

**Lemma 2.2** *Let  $\mathbf{Y}$  be an  $N$  run,  $k$  factor,  $s$ -level factorial design with columns  $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k\}$ . Let  $\mathbf{Y}'$  be the  $N \times sk$  matrix with columns  $\{\mathbf{y}'_1, \mathbf{y}'_2, \dots, \mathbf{y}'_{sk}\}$  where for  $j = 1, \dots, k$  and  $r = 1, \dots, s$  the  $i$ th entry of  $\mathbf{y}'_{s(j-1)+r}$  is 1 if the  $i$ th entry of  $\mathbf{y}_j$  is  $r - 1$  and is 0 otherwise. Then  $\mathbf{Y}$  is an  $OA(N, k, s, t)$  if and only if*

$$\sum_{i=1}^N y'_{ih_1} y'_{ih_2} \cdots y'_{ih_t} = \lambda \quad (2)$$

for any  $t$  columns  $\{\mathbf{y}'_{h_1}, \mathbf{y}'_{h_2}, \dots, \mathbf{y}'_{h_t}\}$  of  $\mathbf{Y}'$  such that  $|h_i - h_j| > s - 1$  for all  $t \geq i > j \geq 1$ .

The result of Lemma 2.2 follows directly from the definition of an  $OA(N, k, s, t)$ .

**Definition 2.3** *Let  $\mathbf{Y}'$  as in Lemma 2.2 be a solution to the system of  $s^t \binom{k}{t}$  equations (2). Also, let  $\mathbf{Y}$  be the  $OA(N, k, s, t)$  obtained from  $\mathbf{Y}'$  with columns  $\sum_{r=1}^s (r - 1) \mathbf{y}'_{s(j-1)+r}$  for each  $j = 1, \dots, k$ . Then  $\mathbf{Y}$  is called the corresponding  $OA(N, k, s, t)$  to  $\mathbf{Y}'$ .*

**Algorithm 2.4 (Identity Group)** *Use basic Algorithm 2.1 with the following ILP extension step.*

1. (a) *Construct  $\mathbf{Y}'_l$  from  $\mathbf{Y}_l$  the same way  $\mathbf{Y}'$  is constructed from  $\mathbf{Y}$  in Lemma 2.2. Extend  $\mathbf{Y}'_l$  with  $s$  columns of binary variables  $\mathbf{x}_r = (x_{1r}, x_{2r}, \dots, x_{Nr})'$  for  $r = 1, \dots, s$ . Now, equations (2) in Lemma 2.2 are a linear system in binary variables.*

(b) Obtain all solutions to the ILP:

$$\begin{aligned}
& \min \quad \mathbf{1}'_N (\sum_{r=1}^{s-1} \mathbf{x}_r) \\
& \text{such that} \quad \sum_{i=1}^N y'_{ih_1} y'_{ih_2} \cdots y'_{ih_{t-1}} x_{ir} = \lambda, \\
& \quad \quad \quad \sum_{r=1}^{s-1} x_{ir} \leq 1, \\
& \mathbf{x}_r \in \{0, 1\}^N, \quad i = 1, \dots, N, \quad r = 1, \dots, s-1
\end{aligned} \tag{3}$$

for any  $t-1$  columns  $\{\mathbf{y}'_{h_1}, \mathbf{y}'_{h_2}, \dots, \mathbf{y}'_{h_{t-1}}\}$  of  $\mathbf{Y}'_l$  with  $1 \leq h_1 < h_2 < \dots < h_{t-1} < sk - s + 1$ , where  $|h_i - h_j| > s - 1$  for all  $t \geq i > j \geq 1$  and  $\mathbf{1}_N$  is an  $N \times 1$  vector of 1s. Each solution  $\mathbf{x}_r$  to ILP (3) has a corresponding  $OA(N, k, s, t)$ . Take  $M_l$  to be the set of all such  $OA(N, k, s, t)$ .

The  $\binom{k-1}{t-1} s^{t-1} (s-1)$  constraints (3) imply that any projection of  $t$  columns which includes the new column will be a  $s^t$  full factorial design with runs replicated  $\lambda$  times. The last column  $\mathbf{x}_s$  of binary variables was deleted as  $\sum_{r=1}^s \mathbf{x}_r = \mathbf{1}_N$ . The dummy objective function  $\mathbf{1}'_N (\sum_{r=1}^{s-1} \mathbf{x}_r)$  was introduced for convenience, so we could simply apply ILP solvers provided by Professor Margot.

Table 1 lists the constraints (3) when extending the  $OA(20,2,2,2)$  to all  $OA(20,3,2,2)$ . The solution set of Algorithm 2.4 Step 1 includes:

$$\mathbf{x}'_1 = (1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0) \tag{4}$$

$$\mathbf{x}'_1 = (1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0) \tag{5}$$

because both satisfy the 4 equality constraints in Table 1 with binary entries. Appending either new column  $\mathbf{1}_N - \mathbf{x}_1$  to the  $OA(20,2,2,2)$  produces an  $OA(20,3,2,2)$ , but these two  $OA(20,3,2,2)$  are isomorphic, i.e., simply permute runs 4 and 5 to go from one of these  $OA(20,3,2,2)$  to the other. In general, permuting entries of a solution vector  $\mathbf{x}_1$  within replicated runs of the input  $OA(N, k-1, s, t)$  will produce a solution vector. Even if distinct, these solutions vectors correspond to isomorphic  $OA(N, k, s, t)$ .

Table 2 contains enumeration results of  $OA(20, k, 2, 2)$  based on the new versions of basic Algorithm 2.1 described in this section. Algorithm 2.4 is inefficient and drastically over-represents designs when  $k$  is small because Step 1 does not have an isomorphism rejection procedure. For example, when  $k = 3$ ,  $|M| = 21, 252$  is much larger than  $|T_3| = 3$  because of the symmetry issue discussed in the last paragraph.

Fixing the first run of the new factor at the low level and ordering factor levels in the new column over replicates within the input design boosts speed by reducing  $|M|$ .

**Algorithm 2.5 (Identity Group with Symmetry Breaking Constraints)** Use Algorithm 2.4 after adding two types of constraints to ILP (3).

- The variable fixing constraint:  $x_{1,1} = 1$ .
- Set(s) of  $s - 1$  symmetry breaking inequality constraint(s) for each pair of replicate runs  $1 \leq i < j \leq N$  in the input design:  $\sum_{m=1}^n x_{i,m} - x_{j,m} \geq 0$  for each  $n = 1, \dots, s - 1$ .

In the Table 1 example, Algorithm 2.5 introduces 41 additional constraints: the variable fixing constraint  $x_{1,1} = 1$  and  $4\binom{5}{2} = 40$  symmetry breaking inequalities  $x_{i,1} - x_{i',1} \geq 0$  for all pairs of replicated runs  $1 \leq i < i' \leq 20$  in the OA(20,2,2,2). Note the uniformly superior performance of Algorithm 2.5 over Algorithm 2.4 displayed in Table 2. The variable fixing constraint alone cuts  $|M|$  in half. Additional reductions are due to the symmetry breaking constraints, but these do not help when  $k$  is large because there are no replicated runs. Note that vector (4) is still an Algorithm 2.5 solution in the Table 1 example, but vector (5) is now rejected by the symmetry breaking constraints.

The isomorphism pruning of Margot (2007) in a branch-and-cut algorithm is formulated below to apply isomorphism rejection while solving the needed ILPs.

**Definition 2.6** For an OA( $N, k, s, t$ ) =  $\mathbf{Y}$ , let  $\mathbf{Y}'$  be the  $N \times sk$  matrix as in Lemma 2.2. Let  $G_s^k = \{\pi \in S_N \mid \text{there exists } \sigma \in S_{sk} \text{ such that } \mathbf{Y}'(\pi, \sigma) = \mathbf{Y}'\}$ , where  $S_m$  is the group of all permutations of  $m$  objects and  $\mathbf{Y}'(\pi, \sigma)$  is the resulting matrix when the rows and columns of  $\mathbf{Y}'$  are permuted with  $\pi$  and  $\sigma$ , respectively. Then  $G_s^k$  is called the automorphism group corresponding to  $\mathbf{Y}$ .

**Algorithm 2.7 (Isomorphism Pruning)** Same as Algorithm 2.4, except variable fixing and isomorphism pruning with automorphism group  $G_s^{k-1}$  corresponding to  $\mathbf{Y}_1$  are used to solve ILP (3).

Algorithm 2.7 always produces a number of solutions  $|M|$  no bigger than that of Algorithm 2.5 because the group used by Algorithm 2.7 takes a solution factorial design to all of its isomorphic copies obtained by either permuting replicated runs in the input design or permuting run indices that send the input design to itself after relabeling the factor levels in a subset of columns. This group, however, is a subgroup of the set of all permutations that takes a solution factorial design to one of its isomorphic copies; for example, Algorithm 2.7 extends the unique OA(20,2,2,2) to  $|M| = 5$  OA(20,3,2,2), not  $|T_3| = 3$ . Algorithm 2.5 is uniformly faster than Algorithm 2.7 in the enumeration of OA(20, $k$ ,2,2) because the use of isomorphism pruning does not overcome its computational overhead in these simple problems. However, with larger  $N$  and moderate  $k$ , Algorithm 2.7 is vastly superior to Algorithm 2.4 because Algorithm 2.7 efficiently produces smaller  $|M|$  as mentioned above. For example, extending OA(24,5,2,2) to OA(24,6,2,2) takes

1.52 and 4.45 minutes with Algorithms 2.7 and 2.4, respectively. This efficiency gap is much wider with larger  $N$  and moderate  $k$ . However, even with large  $N$ , Algorithm 2.5 is most efficient with large  $k$  because isomorphism pruning is not worthwhile when the ILPs have little symmetry.

We observed that  $|G_s^{k-1}|$ , used in Algorithm 2.7, is very large when  $k-1-t$  is small and  $N$  is large. This is mainly due to the fact that for small  $k-t-1$  an  $\text{OA}(N, k-1, s, t)$  may have many rows replicated several times. This causes  $|G_s^{k-1}|$  to be very large as all permutations in  $S_N$  that send a row's index to one of its replicates' indices are in  $G_s^{k-1}$ . When  $|G_s^{k-1}|$  is very large, the Margot (2007) branch-and-cut with isomorphism pruning stalls. For  $s=2$ , we overcame this issue by introducing the symmetry breaking inequalities. Once these inequalities were introduced, the maximal subgroup  $H_2^{k-1}$  of  $G_2^{k-1}$  that does not have an element sending a row's index to one of its replicate rows' index was used.

**Algorithm 2.8 (Isomorphism Pruning with Symmetry Breaking Constraints)** *Same as Algorithm 2.5, except isomorphism pruning with the subgroup  $H_2^{k-1}$  of  $G_2^{k-1}$  is used to solve ILP (3). This algorithm only applies when  $s=2$ .*

Table 2 shows that  $|M|$  are the same for Algorithms 2.7 and 2.8; this is true in general. Furthermore, the times for Algorithms 2.7 and 2.8 are comparable in Table 2 because the large group size issue mentioned above (which motivated us to develop Algorithm 2.7) is not present in these simple  $N=20$  enumerations. Extending all  $\text{OA}(160,8,2,4)$  to all  $\text{OA}(160,9,2,4)$  is an example when this group size issue is present, and furthermore, Algorithm 2.8 was our only version of Algorithm 2.1 which solved the case  $\text{OA}(160,9,2,4)$ .

Algorithms 2.4, 2.5, 2.7, and 2.8 are based on the basic extension Algorithm 2.1 framework; furthermore, they each solve an ILP in Step 1 with  $N(s-1)$  binary variables. On the other hand, the Bulutoglu and Margot (2008) algorithm solves an ILP in  $s^k$  nonnegative integer variables to directly find all  $\text{OA}(N, k, s, t)$ . A shortcoming of the Bulutoglu and Margot (2008) algorithm is that it cannot be easily put in parallel, like Step 1 of a basic extension Algorithm 2.1. We formulate a new basic extension version of the Bulutoglu and Margot (2008) algorithm after a brief introduction to the latter.

The Bulutoglu and Margot (2008) formulation is defined by the  $\binom{k}{t}s^t \times s^k$  constraint matrix  $\mathbf{A}$  with known binary entries and an unknown solution vector  $\mathbf{x} := (x_1, x_2, \dots, x_{s^k})'$ . We now illustrate the definitions of  $\mathbf{A}$  and  $\mathbf{x}$  through the example of enumerating all  $\text{OA}(\lambda 2^2, 3, 2, 2)$ ; see Bulutoglu and Margot (2008) for details. First, obtain the  $2^3$  full factorial design ordered lexicographically by runs:

000	→ Run 1
001	→ Run 2
010	→ Run 3
011	→ Run 4
100	→ Run 5
101	→ Run 6
110	→ Run 7
111	→ Run 8,

and let  $x_i$  be the number of times that the  $i$ th run appears in an  $\text{OA}(\lambda^2, 3, 2, 2)$ . We must have the following  $\binom{k}{t} s^t = 12$  linear equations in  $s^k = 8$  nonnegative integer unknowns which determine the constraint matrix  $\mathbf{A}$ .

Run	Associated Equation
(0, 0, ●)	$x_1 + x_2 = \lambda$
(0, 1, ●)	$x_3 + x_4 = \lambda$
(1, 0, ●)	$x_5 + x_6 = \lambda$
(1, 1, ●)	$x_7 + x_8 = \lambda$
(0, ●, 0)	$x_1 + x_3 = \lambda$
(0, ●, 1)	$x_2 + x_4 = \lambda$
(1, ●, 0)	$x_5 + x_7 = \lambda$
(1, ●, 1)	$x_6 + x_8 = \lambda$
(●, 0, 0)	$x_1 + x_5 = \lambda$
(●, 0, 1)	$x_2 + x_6 = \lambda$
(●, 1, 0)	$x_3 + x_7 = \lambda$
(●, 1, 1)	$x_4 + x_8 = \lambda$

**Algorithm 2.9 (Bulutoglu and Margot, 2008)** *Input:*  $N, k, s, t$ . *Find all solutions to ILP:*

*minimize:*

$$\sum_{i=1}^{s^k} x_i$$

*subject to:*

(6)

$$\mathbf{Ax} = \lambda \mathbf{1},$$

$$x_i \in \{0, \dots, p_{\max}\} \quad \text{for } i = 1, \dots, s^k,$$

using symmetry group  $G_{s,k+1}$  from Bulutoglu and Margot (2008) to prune the enumeration tree. Compute  $p_{\max}$  by solving the ILP in Lemma 5 in Bulutoglu and Margot (2008). *Output:*  $T_k$ .

The group  $G_{s,k+1}$  in ILP (6) maps a solution  $\mathbf{x}$  corresponding to an  $\text{OA}(N, k, s, t)$  to that of an isomorphic OA; this fact implies that the branch-and-cut algorithm with isomorphism pruning returns one  $\text{OA}(N, k, s, t)$  per isomorphism class. Therefore, no isomorphism reduction step, like Step 3 of Algorithm 2.1, is necessary. In addition to the parallel computing issue, the other shortcoming of Algorithm 2.9 mentioned earlier is that the number of variables  $s^k$  is exponential in  $k$ .

Next, a hybrid of Algorithms 2.1 and 2.9 is defined. Its advantages include 1) an  $s^k$  variable formulation within the basic extension Algorithm 2.1 framework so that Step 1 can be put in parallel and 2) a reduction in the number of variables from  $s^k$  to  $hs$ , where  $h \leq N$  is the number of distinct runs in the  $\text{OA}(N, k-1, s, t)$  being extended. This is accomplished by adding constraints to ILP (6) based on the  $\text{OA}(N, k-1, s, t)$ . Consider the example of extending the following  $\text{OA}(16, 4, 2, 3)$  to an  $\text{OA}(16, 5, 2, 3)$ .

$y_1$	$y_2$	$y_3$	$y_4$
0	0	0	0
0	0	0	0
0	0	1	1
0	0	1	1
0	1	0	1
0	1	0	1
0	1	1	0
0	1	1	0
1	0	0	1
1	0	0	1
1	0	1	0
1	0	1	0
1	1	0	0
1	1	0	0
1	1	1	1
1	1	1	1

Rows not in this OA(16,4,2,3) reduce the number of variables from  $s^k$  to  $hs$  as follows.

Rows not in OA(16,4,2,3)	Additional Equations
(0, 0, 0, 1)	$x_3 = 0, x_4 = 0$
(0, 0, 1, 0)	$x_5 = 0, x_6 = 0$
(0, 1, 0, 0)	$x_9 = 0, x_{10} = 0$
(1, 0, 0, 0)	$x_{17} = 0, x_{18} = 0$
(1, 1, 1, 0)	$x_{29} = 0, x_{30} = 0$
(1, 1, 0, 1)	$x_{27} = 0, x_{28} = 0$
(1, 0, 1, 1)	$x_{23} = 0, x_{24} = 0$
(0, 1, 1, 1)	$x_{15} = 0, x_{16} = 0$

We also get one equation per row in this OA(16,4,2,3) as follows.

Extended Row	Additional Equation
(0, 0, 0, 0)	$x_1 + x_2 = 2$
(0, 0, 1, 1)	$x_7 + x_8 = 2$
(0, 1, 0, 1)	$x_{11} + x_{12} = 2$
(0, 1, 1, 0)	$x_{13} + x_{14} = 2$
(1, 0, 0, 1)	$x_{19} + x_{20} = 2$
(1, 0, 1, 0)	$x_{21} + x_{22} = 2$
(1, 1, 0, 0)	$x_{25} + x_{26} = 2$
(1, 1, 1, 1)	$x_{31} + x_{32} = 2$

Here is how these equations are written in general. Let  $i = 1, \dots, s^{k-1}$  index the runs of the  $s^{k-1}$  full factorial design (with runs ordered lexicographically), and  $r_i$  be the replication of run  $i$  in the input OA( $N, k-1, s, t$ ). For each  $r_i = 0$ , add  $s$  constraints  $x_{(i-1)s+j} = 0$  for  $j = 1, \dots, s$ ; for each  $r_i > 0$ , add the constraint  $\sum_{j=1}^s x_{(i-1)s+j} = r_i$ . Extend  $\mathbf{A}$  to  $\tilde{\mathbf{A}}$  with binary rows and  $\mathbf{1}\lambda$  to  $\mathbf{c}$  with  $r_i$  accordingly.

**Algorithm 2.10 (A New Extension Version of Bulutoglu and Margot, 2008)** *Basic Algorithm 2.1 with the following ILP extension Step 1. Find all solutions to ILP (6) after replacing  $\mathbf{A}$  with  $\tilde{\mathbf{A}}$ ,  $\mathbf{1}$  with  $\mathbf{c}$ , and  $G_{s,k}$  with its maximal subgroup  $H_{s,k}$  that maps the input OA( $N, k-1, s, t$ ) to itself.*

An efficiency is that Step 1 of Algorithm 2.10 produces not only “at least 1 representative” but “exactly 1 representative,” so Algorithm 2.10 will never produce an  $|M|$  bigger than those of the other extension algorithms in Table 2.1. As a result, the isomorphism reduction Step 3 of basic Algorithm 2.10 is not needed when  $|T_{k-1}| = 1$ , but is still needed otherwise, e.g., see the 3 and 4 factor rows in Table 2. An

inefficiency is that the number of variables is exponential in  $k$  when there are few replicated runs, so we did not attempt to complete the Algorithm 2.10 column of Table 2 as the speed degradation with increased  $k$  is apparent. On the other hand, while Algorithm 2.10 times reported in Table 2 for small  $k \leq 5$  are only comparable to the faster algorithms, Algorithm 2.10 is orders of magnitude faster than the other extension algorithms in more complicated problems with larger  $N$  and small  $k$ ; this huge boost in speed is clear when extending OA(160,7,2,4) to OA(160,8,2,4) and is discussed further in Section 4.

### 3. DIRECTED SEARCHES FOR GMA OA( $N, k, 2, t$ )

Stufken and Tang (2007) used  $J$ -characteristics to classify all OA( $N, t+2, 2, t$ ). The following generalization of Lemma 3 in Stufken and Tang (2007) is used to find GMA designs when Section 2 enumerations are computationally out of reach due to large  $|T_k|$ .

**Lemma 3.1** *For an OA( $N, k, 2, t$ ) with  $N = \lambda 2^t$  and  $k \geq t+2$  the following hold.*

1. *For any  $l \subseteq \{1, \dots, k\}$ ,  $J_{|l|}(l) = \mu_l 2^t$  for some integer  $\mu_l$ .*
2. *If  $\lambda$  is even, then  $\mu_l$  is even.*
3. *If  $\lambda$  is odd and  $t$  is even, then  $\mu_l$  is odd if  $|l| - t \equiv 1$  or  $2 \pmod{4}$  and even otherwise.*
4. *If  $\lambda$  is odd and  $t$  is odd, then  $\mu_l$  is odd if  $|l| - t \equiv 1$  or  $3 \pmod{4}$  and even otherwise.*

Let  $J_{t+1}(l)$  be a  $J$ -characteristic of an OA( $N, k, 2, t$ ) based on some subset  $l$  of  $t+1$  factors. By Lemma 3.1,  $J_{t+1}(l) = \mu_l 2^t$  for some integer  $\mu_l$ , and  $\mu_l$  is odd if and only if  $\lambda$  is odd. Now, the set of all OA( $N, k, 2, t$ ) such that all  $|J_{t+1}(l)|$  achieve their lower bound of  $2^t$  with odd  $\lambda$  will contain all GMA designs if this set is nonempty; let  $T'_k$  be this subset of  $T_k$ . The following algorithm is needed for odd  $\lambda$  only because you can increase  $t$  by 1 and use Algorithm 2.8 for even  $\lambda$ .

**Algorithm 3.2 (Minimizing All  $|J_{t+1}(l)|$ )** *Use Algorithm 2.7 after replacing  $T_{k-1}$  with  $T'_{k-1}$  and  $T_k$  with  $T'_k$ , and modifying ILP (3).*

- *Add  $2 \binom{k}{t} 2^t$  directed search constraints  $(\lambda - 1)/2 \leq \sum_{i=1}^{\lambda} x_{j_i} \leq (\lambda + 1)/2$ , where a subset of  $\lambda$  runs  $j_i$  in the input design, i.e.,  $1 \leq j_1 < j_2 < \dots < j_{\lambda} \leq N$ , replicates a run in the  $2^t$  full factorial when the input design is projected onto  $t$  columns.*
- *Use group  $G_2^k$  as if extending OA( $N, k-1, 2, t$ ) to OA( $N, k, 2, t$ ) with Algorithm 2.7 for the isomorphism pruning.*

An efficiency of Algorithm 3.2 for small  $k$  is the speed increase due to processing fewer designs; compare Tables 2 and 3. If  $|T'_k| \neq 0$ , the output  $T'_k$  from Algorithm 3.2 is the set of all *weak GMA* designs, where a weak GMA design has the same first non-zero entry in its GWP as a GMA design. Shortcomings of Algorithm 3.2 are a crude lower bound of  $10 \leq f(20, 2, 2)$  and the inability to capture GMA designs for large  $k = 11, \dots, 19 = f(20, 2, 2)$  unlike the full enumerations summarized in Table 2.

Lemmas 3.3 and 3.4 can also be used to restrict the search to a smaller subset of good designs which includes GMA designs if nonempty. Lemma 3.3 follows from Lemma 3.1 and is used in Lemma 3.4; Lemma 3.4 generalizes Lemma 2 in Xu (2009) from regular designs to  $OA(N, k, 2, t)$ .

**Lemma 3.3** *Let  $\mathbf{Y}$  be an  $OA(N, k, 2, t)$ . Then  $[N^2 A_j(\mathbf{Y}) - \phi_1(k, t, j)]/\phi_2(t, j)$  are nonnegative integers for  $j = t + 1, \dots, k$ , where*

$$\begin{aligned}\phi_1(k, t, j) &:= I(t, j) \binom{k}{j} 2^{2t} \\ \phi_2(t, j) &:= 2^{2t+2+I(t, j)}\end{aligned}$$

and  $I(t, j) := 1$  if  $t$  is even and  $j - t \equiv 1$  or  $2 \pmod{4}$  or if  $t$  is odd and  $j - t \equiv 1 \pmod{2}$  and  $I(t, j) := 0$  otherwise.

**Lemma 3.4** *Let  $\mathbf{Y}$  be an  $OA(N, k, 2, t)$  and  $r < k$  be a positive integer. If  $\mathbf{Y}(-i)$  (i.e., design  $\mathbf{Y}$  with the  $i$ th column deleted) is GMA among all other delete-one-factor projections of  $\mathbf{Y}$ , then*

$$A_{t+r}(\mathbf{Y}(-i)) \leq \left\lfloor \frac{N^2(A_{t+r}(\mathbf{Y}) - (t+r)A_{t+r}(\mathbf{Y})/k) - \phi_1(k-1, t, t+r)}{\phi_2(t, t+r)} \right\rfloor / N^2. \quad (7)$$

**Algorithm 3.5 (Minimizing All  $|J_{t+1}(l)|$ , and Recursion (7) on  $A_{t+2}(\mathbf{Y})$ )** *Let  $T''_k$  be the subset of  $T'_k$  satisfying recursion (7), with initial values  $k_{\text{input}} \geq k$  and  $A_{t+2}$  for the right hand side. Replace each “ $q$ ” with “ $n$ ” in Algorithm 3.2 and discard designs not satisfying recursion (7) after solving each ILP.*

Compare Tables 3 and 4 and note another speed increase due to further reduction in the number of designs because of recursion (7). Similarly, Algorithm 3.5 provides an even cruder lower bound of  $7 \leq f(20, 2, 2)$ .

Assume odd  $\lambda$  momentarily. Algorithm 3.2 can be used for given  $(N, s, t)$  to find GMA designs for small and medium  $k$  when the full enumeration methods of Section 2 are computationally too intense. Algorithm 3.5 can be used if Algorithm 3.2 requires too much computation, but then only small  $k$  GMA designs will be obtained. This approach was used to further extend our new catalog of GMA designs; the catalog and its construction are explained next. (Using recursion (7) with odd or even  $\lambda$  to direct

enumerations on designs with good  $A_{t+1}$  is straightforward, but was not applied.)

#### 4. A CATALOG OF GMA $OA(N, k, s, t)$

We used the tools from Sections 2 and 3 to produce an extensive  $OA(N, k, s, t)$  catalog; summaries of this catalog are listed in Tables 5 and 6. Note the huge number of non-isomorphic  $OA(24, k, 2, 2)$  in Table 5 compared to that of  $OA(20, k, 2, 2)$  in Table 2. The enumeration of  $OA(24, k, 2, 2)$  would take more than 3 years if run on an OSC processor. By the calendar, it finished after a few months because of parallel processing. Step 1 of basic Algorithm 2.1 was typically spread over 130 CPUs at OSC whenever there were at least that many extensions and more than a day would have been used on a single computer. Quick jobs and some Step 3 isomorphism reductions of basic Algorithm 2.1 (i.e., those that took more than a week or more than 100GB disk space) were run on the second author’s research computer. The times reported in Table 5 (and elsewhere) each indicate how long a job would take on a single computer, albeit processor speed was not necessarily constant between or within jobs summarized in Table 5.

An important issue was determining the most efficient algorithm to produce a given row in Table 5, or else the job might not have finished. For small  $k$ , Algorithm 2.9 was fast. For the more time consuming problems with moderate and large  $k$ , we typically tested each extension algorithm on the same small subset of  $OA(N, k - 1, s, t)$ , and then used the fastest to finish the full job. With large  $k$ , Algorithm 2.5 was efficient because many of the needed ILPs had little symmetry, so use of isomorphism pruning was unnecessary computational overhead. This, however, was not true for small and moderate  $k$  where the other versions of basic Algorithm 2.1 which use isomorphism pruning were essential as explained previously in Section 2. Enumerating  $OA(160, 8, 2, 4)$  took 540 hours with Algorithm 2.10. When spread over 130 CPUs, the “effective time” was 47 hours for the slowest of the 130 sets of extensions plus 5 more hours for the isomorphism reduction. This effective time is 11.1 times faster than the 522 hours this job took using Algorithm 2.9. We consistently observed that on a single machine that Algorithm 2.10 is a little slower than Algorithm 2.9, but we have no method for parallel processing with Algorithm 2.9.

The status column in Table 5 indicates two pieces of information. First, if an abbreviated reference appears, then a GMA design was previously established by that source. Second, if an asterisk is next to a reference, then a GMA design was obtained via a construction with theoretical proof; whereas a reference with no asterisk means that a GMA design was obtained via enumeration of  $OA(N, k, s, t)$ .

The 28 and 36 runs cases were directed searches. With  $N = 28$ , Algorithm 3.2 was used, so the number of classes in Table 5 is  $|T'_k|$ . With  $N = 36$ , Algorithm 3.5 was used, so the number of classes in

Table 5 is  $|T_k''|$ , where the initial values for recursion (7) were  $k_{\text{input}} = 18$  and  $A_4 = 120,000/36^2$ . While some discussion on choosing initial values to recursion (7) is given later in Section 6, these particular values were picked using trial-and-error to produce a directed search which would finish reasonably fast and also produce new designs. In all other cases throughout Table 5, the number of classes is  $|T_k|$ .

All of the Section 2 enumeration algorithms except Algorithm 2.8 apply for general  $s$ . However, with  $s = 3$ , additional variable fixing constraints were used to force new factors to start with 000, 001, 010, 011, and 012 per the results of Evangelaras *et al.* (2009). With these constraints, we quickly reproduced an enumeration of  $\text{OA}(27, k, 3, 2)$  and matched the GMA designs found in Evangelaras *et al.* (2009), but note their typo  $|T_{13}| = 129$ . We obtained  $|T_{13}| = 68$  which agrees with the constructions of Lam and Tonchev (1996). Our methods also made quick work of the  $\text{OA}(81, k, 3, 3)$  enumeration; this new enumeration result is given in Tables 5 and 6. Xu (2005b) enumerated all 3-level regular designs with 27 and 81 runs. We confirm that each of the MA regular designs with 27 runs are GMA and that each of the MA designs with 81 runs and  $k \leq 10 = f(81, 3, 3)$  are GMA. An enumeration of  $\text{OA}(81, k, 3, 2)$  might be used to prove or disprove the GMA status of MA designs with 81 runs when  $k > 10$ . Xu (2005b) was not cited in the status column of Table 5 for the 81 runs cases, because an MA design is not necessarily GMA, e.g., see Xu (2005). We note that more variable fixing refinements should be added to our methods before enumerating difficult cases  $\text{OA}(N, k, s, t)$  with  $s > 3$ .

Our three main contributions are stated. 1) We obtained a large catalog of GMA designs with small  $N$  for practical use. For a GMA design, the practitioner is referred to the first author's webpage. 2) Our catalog may also be useful to theorists, who derive lower bounds for the GWP. They can compare the corresponding distance distribution of their lower bound to Table 6. We also note that in cases where the GMA status in Table 5 has a reference, our computational results agree. 3) Those who develop algorithms have a basis to test the correctness and to compare the speed of new methods using Tables 5 and 6.

## 5. QUICK HEURISTIC SEARCH FOR NEAR GMA $\text{OA}(N, k, s, t)$

Two search algorithms for quickly producing weak and near GMA designs (that are not necessarily GMA) are described. The effectiveness of these algorithms is demonstrated by re-locating known GMA designs with the properties listed in Table 6. Then, the search algorithms are used to locate weak and near GMA  $\text{OA}(N, k, s, t)$  when enumeration and directed search approaches involve too much computation. All jobs in this section were run on the 3GHz research computer mentioned earlier.

Given an  $\text{OA}(N, k_0, s, t)$ , the following search optimizes the GWP of a randomly selected subset of

$k < k_0$  columns by changing this subset an-element-at-a-time to maximize reductions in GWP.

**Algorithm 5.1 (Search Based on Projections)** *Input:* An  $OA(N, k_0, s, t)$  say  $\mathbf{Y}$ ,  $k < k_0$ , and a number iterations  $R$ . Repeat the following  $R$  times.

1. Randomly select a projection  $\mathbf{Y}_p$  of  $k$  columns from  $\mathbf{Y}$ . Let  $\mathbf{Y}_p^c$  be the  $N \times k_0 - k$  complement of  $\mathbf{Y}$  wrt  $\mathbf{Y}_p$ , i.e., the columns of  $\mathbf{Y}$  not in  $\mathbf{Y}_p$ . Set  $j := 1$  and  $\text{count} := 0$ .
2. Compute the  $k_0 - k$  changes in the GWP when column  $j$  of  $\mathbf{Y}_p$  is replaced with each column of  $\mathbf{Y}_p^c$ . If one of these column swaps reduces GWP, pick a swap which reduces GWP the most, update  $\mathbf{Y}_p$  and  $\mathbf{Y}_p^c$  accordingly, and reset  $\text{count} := 0$ ; else increment  $\text{count} := \text{count} + 1$ .
3. If  $\text{count} < k$ , set  $j := \max\{(j + 1) \bmod (k + 1), 1\}$  and GOTO Step 2.
4. The projection  $\mathbf{Y}_p$  is locally optimal. If this is the first iteration, store  $\mathbf{Y}_p$  and its GWP; else if the current GWP is an improvement, overwrite with  $\mathbf{Y}_p$  and its GWP.

*Output:* A  $k$  column projection  $\mathbf{Y}_p$  of  $\mathbf{Y}$  and the GWP of  $\mathbf{Y}_p$ .

We obtained inputs  $OA(N, k_0, s, t)$  to test Algorithm 5.1 from Neil Sloane's web catalog of orthogonal arrays and Hadamard matrices. An  $N \times N$  Hadamard matrix  $\mathbf{H}_N$ , i.e., an orthogonal matrix with entries  $\pm 1$ , can be used to obtain  $OA(N, N - 1, 2, 2)$  as follows. Normalize  $\mathbf{H}_N$  (by multiplying each column of  $\mathbf{H}_N$  element-wise with a column of  $\mathbf{H}_N$ ), delete the constant column, and recode with binary entries. Thus,  $N$   $OA(N, N - 1, 2, 2)$  can be obtained from a  $\mathbf{H}_N$  in this manner. Doing this once for each Hadamard inequivalent  $\mathbf{H}_N$  and removing isomorphic copies produces all non-isomorphic  $OA(N, N - 1, 2, 2)$ . Table 7 lists the number of  $OA(N, k_0, s, t)$  we tried. All nonequivalent Hadamard matrices of order 28 or less and the construction described above were used to produce more  $OA(N, N - 1, 2, 2)$ . Out of convenience, we only included the 2 available Hadamard matrices of order 36 and 1 of each of the available  $OA(40, 20, 2, 3)$ ,  $OA(48, 24, 2, 3)$ ,  $OA(81, 40, 3, 2)$ , and  $OA(160, 80, 2, 3)$ .

Algorithm 5.1 does not exhaustively consider all  $\binom{k_0}{k}$  projections of  $\mathbf{Y}$  onto  $k$  columns, so an optimal projection is not necessarily found. However, our experience is that Algorithm 5.2 finds an optimal projection even with a seemingly small number of random projections  $R$  and is fast. For example, Algorithm 5.1 was run for each  $k = 3, \dots, 23$  and each  $OA(24, 23, 2, 2)$  with  $R = 100$ ; in total, this took 2 minutes and produced a GMA  $OA(24, k, 2, 2)$  for each  $k = 3, \dots, 23$ . In comparison, the  $OA(24, k, 2, 2)$  enumeration of Section 4 took 2.7 years, but only techniques from Section 4 are guaranteed to produce a design with the best possible GWP. In fact, we used Algorithm 5.1 to find a GMA design in 84 of the 99

cases  $OA(N, k, s, t)$  listed in Table 6 in 2 hours; the calculation was calls of Algorithm 5.1 with  $R = 100$  for each combination of  $(N, k, s)$  in Table 6 and each input  $OA(N, k_0, s, t)$  from Table 7.

A GMA design may not be embedded in any member of a set of input designs to Algorithm 5.1. The “hidden” GMA  $OA(28, k, 2, 2)$  with  $k = 10, 11, 12$  are not a projection of any  $OA(28, 27, 2, 2)$ . Table 8 lists the  $99 - 84 = 15$  non-located GMA  $OA(N, k, s, t)$  and classes of input designs to Algorithm 5.1 which have no projection onto  $k$  columns that is globally GMA. These results support the need to develop search algorithms not based on backwards searches of known  $OA(N, k_0, s, t)$ . The following algorithm incorporates an ILP based forward search to extend into hidden GMA  $OA(N, k, s, t)$ .

**Algorithm 5.2 (ILP Based Search)** *Input:*  $N, t, s$ , the  $OA(N, t, s, t)$ , and  $\text{last} \geq 1$ . Set  $k_{\max} := t$  and  $\text{current} := 1$ .

1. If  $\text{current} \leq \text{last}$ , set  $k := t + 1$  and  $\mathbf{Y}$  to the  $OA(N, t, s, t)$  and GOTO Step 2; else exit.
2. Randomly select  $s - 1$  binary sequences each of length  $N$  with  $N/s$  ones; for each  $r = 1, \dots, s - 1$ , call these sequences  $c_{rj}$ , where  $j = 1, \dots, N$ . Replace the objective function from an ILP in Section 2 or Section 3 with  $\sum_{r=1}^{s-1} \sum_{j=1}^N c_{rj} x_{rj}$ . Find a solution to the resulting ILP or prove infeasibility using a branch-and-cut algorithm. If the ILP is infeasible, increment  $\text{current} := \text{current} + 1$  and GOTO Step 1; else set  $\mathbf{Y}$  to the corresponding  $OA(N, k, s, t)$  and GOTO Step 3.
3. If  $k_{\max} < k$ , then increment  $k_{\max} := k_{\max} + 1$  and store  $\mathbf{Y}$ ; else overwrite the previously stored  $OA(N, k, s, t)$  with  $\mathbf{Y}$  if  $\mathbf{Y}$  is superior under the GMA criterion. If  $\mathbf{Y}$  was stored and  $k > t + 1$ , GOTO the next step with  $\mathbf{D} := \mathbf{Y}$  and  $j := k$ ; else increment  $k := k + 1$  and GOTO Step 2.
4. Obtain  $\mathbf{D}(-i^*)$ , a GMA delete-one-projection  $\mathbf{D}(-i)$  (for  $i = 1, \dots, j$ ) of  $\mathbf{D}$ . If  $\mathbf{D}(-i^*)$  is also superior to the previously stored  $OA(N, j - 1, s, t)$ , store  $\mathbf{D}(-i^*)$  instead, set  $\mathbf{D} := \mathbf{D}(-i^*)$  and  $j := j - 1$ , and then repeat this step if  $j > t + 1$ ; otherwise increment  $k := k + 1$  and continue to randomly extend  $\mathbf{Y}$  by going back to Step 2.

*Output:*  $k_{\max}$  and the  $OA(N, k, s, t)$  with the best found GWP for each  $k \leq k_{\max}$ .

The output  $k_{\max}$  is a lower bound for  $f(N, s, t)$ . Step 2 is a forward selection. When  $s = 2$ , only 1 random sequence  $c_{1j}$  is obtained; in this case, the  $N/2$  row indices  $j$  such that  $c_{1j} = 1$  correspond to a “proposed column,” i.e., these rows are set to the high level and all others the low level. If the proposed column corresponds to a feasible solution of the ILP, the proposed column will correspond to the unique solution of the ILP; otherwise if  $\mathbf{Y}$  can be extended to an  $OA(N, k, s, t)$  subject to the constraints of the

ILP being used, a column which is as close as possible to the proposed column will be used to extend  $\mathbf{Y}$ . Step 4 of Algorithm 5.2 is a backwards selection based on exhaustive search of delete-one-projections. Equation (1) was used to compute components of GWP, and when a new design was stored, its GWP was also stored to avoid duplication of work.

Algorithm 5.2 was run with  $\text{last} = 1,000,000$  for each  $(N, s, t)$  from Section 4; a job was killed after all known GMA designs were found or 1 week of computation. The results in Table 9 show that 78 of the 99 GMA  $\text{OA}(N, k, s, t)$  listed in Table 6 were found. The forward search was able to find 8 of the hidden GMA  $\text{OA}(N, k, s, t)$  listed in Table 8. The speed of Algorithm 5.2 (measured by Current over Time) degrades with increased  $N$ , but increases with increased  $t$  or use of the directed search constraints. We note that  $k_{\max} = 8 < 10 = f(81, 3, 3)$ , but for the other  $(N, s, t)$ , except the  $N = 36$  case, the lower bounds  $k_{\max}$  all achieve the largest possible number of factors when compared with Table 5. With  $(N, s, t) = (36, 2, 2)$ ,  $k_{\max} = 16 > 9$  (where 9 is the largest  $k$  of a GMA  $\text{OA}(36, k, 2, 2)$  in Table 6) because recursion (7) was not used here. Furthermore, any captured  $\text{OA}(36, k, 2, 2)$  is weak GMA because the directed search constraints of Algorithm 3.2 minimize  $A_3$  as mentioned previously in Section 3.

Table 10 has partial GWPs of the best  $\text{OA}(36, k, 2, 2)$  obtained from the search algorithms. Algorithm 5.1 was run with  $R = 1,000$  for each of the 2 input  $\text{OA}(36, 35, 2, 2)$  we tried and each  $k = 3, \dots, 19$  and finished in 6 minutes; Algorithm 5.2 was run with  $(N, s, t) = (36, 2, 3)$  as explained in the last paragraph. The obtained designs with  $k \leq 16$  are weak GMA because they are at least as good as a design found with the constraints of Algorithm 3.2; in other words, each of the  $\binom{k}{3}$  absolute characteristics  $|J_3(l)|$  in any of these designs attains the smallest possible value of 4 from Lemma 3.1. Designs with  $k \leq 6$  were not provided because comparison with Table 6 showed that these designs are GMA. However, comparisons of  $A_4$  to those in Table 5 when  $7 \leq k \leq 9$  reveal that the search algorithms found weak GMA designs which are not GMA. The 17 and 18 factor designs are also weak GMA because both achieve the lowest possible values of  $A_3 \geq \binom{17}{3}4^2/36^2 \approx 8.40$  and  $A_3 \geq \binom{18}{3}4^2/36^2 \approx 10.07$ , respectively. Although the same is not true for the design with 19 factors, this does not prove that the 19 factor design is not weak GMA. Furthermore, it is unknown to us which if any of our best found  $\text{OA}(36, k, 2, 2)$  with  $k > 9$  are GMA.

Algorithm 5.1 was run with  $R = 100$  for each of the 7,570 non-isomorphic  $\text{OA}(28, 27, 2, 2)$  and  $k = 15, \dots, 26$ ; this set of jobs took 10 hours. Algorithm 5.2 was re-run with  $(N, s, t) = (28, 2, 2)$  as explained previously, but the Algorithm 2.7 ILP was used this time to attain larger  $k_{\max}$ . This job ran for the full week and outputted  $k_{\max} = 27 = f(28, 2, 2)$ , but better designs were not found after 1 day. Table 11 has partial GWPs of the best  $\text{OA}(28, k, 2, 2)$  with  $k = 14-27$ ; the GMA and weak GMA status of designs with  $15 \leq k \leq 26$  is unknown. The solution to Research Question 7.38 in Hedayat, Sloane, and Stufken (1999)

is 7,570 non-isomorphic  $OA(4\lambda, 4\lambda - 1, 2, 2)$  when  $\lambda = 7$ . The designs, summarized in Tables 10 and 11, and their distance distributions are also provided on the first author's webpage.

## 6. SUMMARY

An extensive catalog of GMA, weak GMA, and near GMA  $OA(N, k, s, t)$  was provided for practical use. Advantages of this catalog over MA regular designs include: 1)  $OA(N, k, s, t)$  do not have the restriction that the sample size is a power of the number of levels and 2) a GMA design is at least as good as an MA design when a regular design exists. Efficient techniques for enumerating  $OA(N, k, s, t)$  were used to obtain the GMA designs with properties listed in Table 5 when  $N = 24, 32, 40, 48, 81,$  and 160. The fundamental problem with enumerating  $OA(N, k, s, t)$  is that the number of non-isomorphic  $OA(N, k, s, t)$  grows exponentially with  $N$ . Directed searches reduced the number of designs and helped find more GMA designs; see the 28 and 36 run cases in Table 5. The heuristic searches in Section 5 quickly found GMA  $OA(N, k, s, t)$ , matching 92 of the 99 distance distributions of GMA designs given in Table 6. Suppose that a heuristic finds a near GMA  $OA(N, k_{\text{input}}, 2, t)$ . A lean directed search based on the near GMA design would find all GMA  $OA(N, k_{\text{input}}, 2, t)$  in theory. However, the number of non-isomorphic designs still grows exponentially in  $N$  with moderate  $k_{\text{input}}$ , so even a directed search may require too much computation. So we reported weak GMA  $OA(36, k, 2, 2)$  and near GMA  $OA(28, k, 2, 2)$  for large  $k$  obtained from fast heuristics, because of computational limits.

## Acknowledgements

The authors are indebted to Professor Francois Margot for providing executable files for his ILP solvers and addressing their questions. As in Ryan and Bulutoglu (2009), the authors again thank Professor Brendan McKay for answering programming questions regarding nauty and the Office of the CIO at Bowling Green State University for developing a research system capable of processing very large files. The programming tips and system made isomorphism reductions of large sets of designs much easier and were necessary background components for obtaining many of the GMA designs in the new catalog.

This work was supported by a grant from the United States Air Force Office of Scientific Research. This work was also supported in part by an allocation of computing time from the Ohio Supercomputer Center. The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the US Government.

## REFERENCES

- Angelopoulos, P., Evangelaras, H., Koukouvinos, C., and Lappas, E. (2007). An effective step-down algorithm for the construction and the identification of nonisomorphic orthogonal arrays. *Metrika*, **66**, 139-149.
- Angelopoulos, P., Koukouvinos, C., and Lappas, E. (2009). On the construction, classification and evaluation of certain two level nonisomorphic orthogonal arrays. *International Journal of Applied Mathematics and Statistics*, **15**, 63-72.
- Applegate, D., Cook, W., Dash, S. and Espinoza, D.G. (2007). Exact solutions to linear programming problems. *OR Letters* **35**, 693-699.
- Bulutoglu, D.A. and Kaziska, D.M. (2009). Improved WLP and GWP lower bounds based on exact integer programming. (submitted)
- Bulutoglu, D. A. and Margot, F. (2008). Classification of orthogonal arrays by integer programming. *J. Statist. Plann. Inference* **138**, 654-656.
- Butler, N.A. (2004). Minimum  $G_2$ -aberration properties of two-level foldover designs. *Statistics & Probability Letters* **67** 121-132.
- Butler, N.A. (2003). Minimum aberration construction results for nonregular two-level fractional factorial designs. *Biometrika* **90** 891-898.
- Cheng, C. S. (1980). Orthogonal arrays with variable numbers of symbols. *Ann. Statist.* **8**, 447-453.
- Espinoza, D. G. (2006). On linear programming, integer programming and cutting planes. PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology.
- Evangelaras, H., Koukouvinos, C. and Lappas, E. (2007). Further contributions to nonisomorphic two level orthogonal arrays. *J. Statist. Plann. Inference* **137**, 2080-2086.
- Evangelaras, H., Koukouvinos, C. and Lappas, E. (2009). 27-run nonisomorphic three level orthogonal arrays: identification, evaluation and projection properties. *Utilitas Mathematica*, in press.
- Hedayat, A., Seiden, E. and Stufken, J. (1997). On the maximal number of factors and the enumeration of 3-symbol orthogonal arrays of strength 3 and index 2. *J. Statist. Plann. Inference* **58**, 43-63.
- Hedayat, A., Sloane, N. J. A. and Stufken, J. *Orthogonal Arrays: Theory and Applications*. New York:

- Springer-Verlag (1999).
- Hedayat, A., Stufken, J. and Su, G. (1997). On the construction and existence of orthogonal arrays with three levels and indexes 1 and 2. *Ann. Statist.* **25**, 2044-2053.
- Lam, C. and Tonchev, V. D. (1996). Classification of affine resolvable 2-(27,9,4) designs. *J. Statist. Plann. Inference* **56**, 187-202.
- Margot, F. (2007). Symmetric ILP: Coloring and small integers. *Discrete Optim.*, **4**, 40-62.
- Margot, F. (2003a). Exploiting orbits in symmetric ILP. *Math. Program., Ser. B* **98**, 3-21.
- Margot, F. (2003b). Small covering designs by branch-and-cut. *Math. Program., Ser. B* **94**, 207-220.
- Margot, F. (2002). Pruning by isomorphism in branch-and-cut. *Math. Program., Ser. A* **94**, 71-90.
- McKay, B. (2007), *Nauty Users's Guide (Version 2.4)* Computer Science Department, Australian National University, Canberra.
- Mukerjee, R. (1982). Universal optimality of fractional factorial plans derivable through orthogonal arrays. *Calcutta Statist. Assoc. Bull.* **31**, 6-68.
- Padberg, M. W. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large scale symmetric traveling salesman problems. *SIAM Review* **33** (1991), 60-100.
- Ryan, K. J. and Bulutoglu, D. A. (2009). Minimum aberration fractional factorial designs with large  $N$ . *Technometrics*, submitted.
- Seiden, E. and Zemach, R. (1966). On orthogonal arrays. *Ann. Math. Statist.* **37**, 1355-1370.
- Stufken, J. and Tang, B. (2007). Complete enumeration of two-level orthogonal arrays of strength  $d$  with  $d + 2$  constraints. *Ann. Statist.* **35**, 798-814.
- Sun, D. X., Li, W. and Ye, K. Q. An algorithm for sequentially constructing non-isomorphic orthogonal designs and its applications. *SUNYYSB Preprint, AMS-02-13* (2004).
- Tang, B. and Deng, L.Y. (1999). Minimum  $G_2$ -aberration for nonregular fractional factorial designs. *Ann. Statist.* **27** 1914-1926.
- Wu, C. F. J. and Hamada, M. *Experiments Planning Analysis and Parameter Optimization*. New York: Wiley (2000).
- Xu, H. (2005). Some nonregular designs from the Nordstrom-Robinson code and their statistical proper-

ties. *Biometrika*, **92** 385-397.

Xu, H. (2005b) A catalogue of three-level regular fractional factorial designs. *Metrika*, **62** 259-281.

Xu, H. (2009). Algorithmic Construction of Efficient Fractional Factorial Designs with Large Run Sizes. *Technometrics*, in press.

Xu, H. and Wu, C. F. J. (2001). Generalized minimum aberration for asymmetrical fractional factorial designs. *Ann. Statist.* **25**, 549-560.

## APPENDIX A: THEORETICAL JUSTIFICATIONS

### A.1 Proof that the Output of Algorithm 2.7 is $T_k$

Each element of  $M$  from Algorithm 2.7 is an  $OA(N, k, s, t)$  by Lemma 2.2, and the set  $M$  is reduced with nauty to contain no isomorphic designs. Let Algorithm 2.7 be implemented with an  $OA(N, k - 1, s, t)$ ,  $\mathbf{Y}_1$ . It suffices to show that the set  $M_1$  contains at least one copy of each non-isomorphic extension of  $\mathbf{Y}_1$  to an  $OA(N, k, s, t)$ . For each  $g \in G_s^{k-1}$ ,  $g\mathbf{Y}'_1$  corresponds to an  $OA(N, k, s, t)$  which is isomorphic to  $\mathbf{Y}_1$ . Next, let  $\mathbf{Y}_1^g$  denote this  $OA(N, k, s, t)$ . Then each solution  $\mathbf{x}$  of the ILP solved in Step 1 corresponds to an extension of  $\mathbf{Y}_1$  that is an  $OA(N, k, s, t)$ . Let  $[\mathbf{Y}_1\mathbf{y}]$  be this extension. Then  $g\mathbf{x}$  also solves this ILP and corresponds to  $[\mathbf{Y}_1^g\mathbf{y}]$ . Since  $[\mathbf{Y}_1\mathbf{y}]$  and  $[\mathbf{Y}_1^g\mathbf{y}]$  are isomorphic the claim is established.

### A.2 Proof that the Output of Algorithm 2.8 is $T_k$

All elements in  $G_2^{k-1}$  are viable to use in isomorphism pruning, except those that permute variables which extend replications of a run in the  $OA(N, k - 1, 2, t)$ . Such permutations are deleted from  $G_2^{k-1}$  to form  $H_2^{k-1}$ ; otherwise a feasible solution will map to an infeasible solution due to the symmetry breaking inequalities. Let  $i = 1, \dots, 2^{k-1}$  index the runs of the  $2^{k-1}$  full factorial design and  $r_i$  be its replication in the  $OA(N, k - 1, s, t)$ . Note that

$$|G_2^{k-1}| = |H_2^{k-1}| \prod_{i=1}^{2^{k-1}} r_i!$$

### A.3 Proof that the Output of Algorithm 2.10 is $T_k$

The indicator variables for the ILP with  $s^k$  variables for finding all  $OA(N, k, s, t)$  are indexed with the  $s^k$  full factorial design. The action of  $G_{s,k}$  on the full factorial design defines the action of  $G_{s,k}$  on the indicator variables. All elements in  $G_{s,k}$  are viable to use in isomorphism pruning, except those that produce a different  $OA(N, k - 1, s, t)$  input design when  $G_{s,k}$ 's action is restricted to the first  $k - 1$  factors.

This is true since the elements that are not viable send a feasible solution (i.e., a feasible extension of the input design) to an infeasible solution (i.e., a design whose first  $k - 1$  factors are not fixed to the input design). The elements that are viable to use in isomorphism pruning send a run that appears  $r_i$  times in the input design to a run that appears the same number of times. These elements form the group  $H_{s,k}$ .

#### A.4 Proof of Lemma 3.4

By Lemma 3.3, it suffices to show  $A_{t+r}(\mathbf{Y}(-i)) \leq A_{t+r}(\mathbf{Y}) - (t+r)A_{t+r}(\mathbf{Y})/k$  for some factor  $i$ . Each  $J_{t+r}(l)$  involves  $t+r$  factors, so on average each factor contributes  $(t+r)A_{t+r}/k$  to  $A_{t+r}$ . Then there must exist a factor that contributes at least  $(t+r)A_{t+r}/k$  to  $A_{t+r}$ . Let this factor be factor  $i$ . Then  $A_{t+r}(\mathbf{Y}(-i)) \leq A_{t+r}(\mathbf{Y}) - (t+r)A_{t+r}(\mathbf{Y})/k$ .

### APPENDIX B: TABLES

Table 1: Extending the OA(20,2,2,2) to OA(20,3,2,2). (The operator  $\odot$  is the element-wise product.)

OA(20,2,2,2)	$\mathbf{y}'_1$	$\mathbf{y}'_2$	$\mathbf{y}'_3$	$\mathbf{y}'_4$	$\mathbf{y}'_1 \odot \mathbf{x}_1$	$\mathbf{y}'_2 \odot \mathbf{x}_1$	$\mathbf{y}'_3 \odot \mathbf{x}_1$	$\mathbf{y}'_4 \odot \mathbf{x}_1$
00	0	1	0	1	0 * $x_{1,1}$	1 * $x_{1,1}$	0 * $x_{1,1}$	1 * $x_{1,1}$
00	0	1	0	1	0 * $x_{2,1}$	1 * $x_{2,1}$	0 * $x_{2,1}$	1 * $x_{2,1}$
00	0	1	0	1	0 * $x_{3,1}$	1 * $x_{3,1}$	0 * $x_{3,1}$	1 * $x_{3,1}$
00	0	1	0	1	0 * $x_{4,1}$	1 * $x_{4,1}$	0 * $x_{4,1}$	1 * $x_{4,1}$
00	0	1	0	1	0 * $x_{5,1}$	1 * $x_{5,1}$	0 * $x_{5,1}$	1 * $x_{5,1}$
01	0	1	1	0	0 * $x_{6,1}$	1 * $x_{6,1}$	1 * $x_{6,1}$	0 * $x_{6,1}$
01	0	1	1	0	0 * $x_{7,1}$	1 * $x_{7,1}$	1 * $x_{7,1}$	0 * $x_{7,1}$
01	0	1	1	0	0 * $x_{8,1}$	1 * $x_{8,1}$	1 * $x_{8,1}$	0 * $x_{8,1}$
01	0	1	1	0	0 * $x_{9,1}$	1 * $x_{9,1}$	1 * $x_{9,1}$	0 * $x_{9,1}$
01	0	1	1	0	0 * $x_{10,1}$	1 * $x_{10,1}$	1 * $x_{10,1}$	0 * $x_{10,1}$
10	1	0	0	1	1 * $x_{11,1}$	0 * $x_{11,1}$	0 * $x_{11,1}$	1 * $x_{11,1}$
10	1	0	0	1	1 * $x_{12,1}$	0 * $x_{12,1}$	0 * $x_{12,1}$	1 * $x_{12,1}$
10	1	0	0	1	1 * $x_{13,1}$	0 * $x_{13,1}$	0 * $x_{13,1}$	1 * $x_{13,1}$
10	1	0	0	1	1 * $x_{14,1}$	0 * $x_{14,1}$	0 * $x_{14,1}$	1 * $x_{14,1}$
10	1	0	0	1	1 * $x_{15,1}$	0 * $x_{15,1}$	0 * $x_{15,1}$	1 * $x_{15,1}$
11	1	0	1	0	1 * $x_{16,1}$	0 * $x_{16,1}$	1 * $x_{16,1}$	0 * $x_{16,1}$
11	1	0	1	0	1 * $x_{17,1}$	0 * $x_{17,1}$	1 * $x_{17,1}$	0 * $x_{17,1}$
11	1	0	1	0	1 * $x_{18,1}$	0 * $x_{18,1}$	1 * $x_{18,1}$	0 * $x_{18,1}$
11	1	0	1	0	1 * $x_{19,1}$	0 * $x_{19,1}$	1 * $x_{19,1}$	0 * $x_{19,1}$
11	1	0	1	0	1 * $x_{20,1}$	0 * $x_{20,1}$	1 * $x_{20,1}$	0 * $x_{20,1}$
Total:					5	5	5	5

Table 5: Enumeration results: the number of isomorphism classes, real time (hours:minutes:seconds), algorithm, status in the literature, and partial GWP of GMA designs.

OA( $N, k, s, t$ )	Classes	Time	Algorithm	Status	$A_{t+1}$	$A_{t+2}$	$A_{t+3}$	$A_{t+4}$
OA(24,3,2,2)	4	00:00:01	2.9	S.&T.	0.00			
OA(24,4,2,2)	10	00:00:01	2.9	S.&T.	0.00	0.11		
OA(24,5,2,2)	63	00:00:05	2.7	A., E., K, &L	0.00	0.56	0.00	
OA(24,6,2,2)	1350	00:01:31	2.7	A., E., K, &L	0.00	1.67	0.00	0.00
OA(24,7,2,2)	57389	01:33:02	2.7	A., E., K, &L	0.00	3.89	0.00	0.44
OA(24,8,2,2)	1470157	55:00:01	2.7	A., K, &L	0.00	7.78	0.00	1.78

Continued on next page

Table 5 – continued from previous page

$OA(N, k, s, t)$	Classes	Time	Algorithm	Status	$A_{t+1}$	$A_{t+2}$	$A_{t+3}$	$A_{t+4}$
OA(24,9,2,2)	12952435	700:25:52	2.7	B. (2004)*	0.00	14.00	0.00	5.33
OA(24,10,2,2)	38592861	3299:09:34	2.7	B. (2003)*	0.00	23.33	0.00	13.33
OA(24,11,2,2)	52912678	2801:15:25	2.5	B. (2003)*	0.00	36.67	0.00	29.33
OA(24,12,2,2)	51154497	2710:12:22	2.5	B. (2003)*	0.00	55.00	0.00	58.67
OA(24,13,2,2)	43092737	5192:47:52	2.5	B. (2003)*	6.00	55.00	40.00	58.67
OA(24,14,2,2)	31833387	4431:17:27	2.5	B. (2003)*	12.00	61.00	80.00	98.67
OA(24,15,2,2)	19960039	2208:23:39	2.5		18.11	73.00	125.33	178.67
OA(24,16,2,2)	10351396	904:40:57	2.5		24.44	91.11	181.33	304.00
OA(24,17,2,2)	4385567	643:15:47	2.5		31.11	115.11	254.22	487.11
OA(24,18,2,2)	1502242	236:45:36	2.5		37.33	148.00	352.00	730.67
OA(24,19,2,2)	409478	73:28:43	2.5		45.33	185.33	472.00	1082.67
OA(24,20,2,2)	86725	14:31:44	2.5	B. (2003)*	53.33	231.67	629.33	1546.67
OA(24,21,2,2)	13833	03:07:49	2.5	B. (2003)*	63.33	285.00	816.00	2176.00
OA(24,22,2,2)	1604	00:28:42	2.5	B. (2003)*	73.33	348.33	1056.00	2992.00
OA(24,23,2,2)	130	00:02:48	2.5	B. (2003)*	84.33	421.67	1349.33	4048.00
OA(24,24,2,2)	0	00:00:02	2.5					
OA(28,3,2,2')	1	00:00:03	3.2	S.&T.	0.02			
OA(28,4,2,2')	3	00:00:01	3.2	S.&T.	0.08	0.02		
OA(28,5,2,2')	15	00:00:02	3.2	A., E., K, &L	0.20	0.10	0.00	
OA(28,6,2,2')	320	00:00:36	3.2	A., E., K, &L	0.41	0.31	0.00	0.73
OA(28,7,2,2')	12194	00:39:14	3.2	A., K, &L	0.71	0.88	1.55	0.41
OA(28,8,2,2')	63606	07:16:06	3.2		1.14	2.90	3.27	0.65
OA(28,9,2,2')	20552	08:02:19	3.2		1.71	5.51	6.61	2.45
OA(28,10,2,2')	841	01:27:38	3.2		2.45	10.49	11.43	5.06
OA(28,11,2,2')	45	00:03:10	3.2		3.37	18.82	14.86	14.69
OA(28,12,2,2')	10	00:00:12	3.2		4.49	28.22	25.47	29.39
OA(28,13,2,2')	2	00:00:02	3.2		5.84	46.43	32.82	59.43
OA(28,14,2,2')	1	00:00:01	3.2		7.43	65.00	52.00	104.00
OA(28,15,2,2')	0	00:00:01	3.2					
OA(32,5,2,3)	5	00:00:01	2.9	S.&T.	0.00	0.00		
OA(32,6,2,3)	10	00:00:04	2.9	A., E., K, &L	0.00	0.00	1.00	
OA(32,7,2,3)	17	00:00:05	2.9	B.&M.	1.00	2.00	0.00	0.00
OA(32,8,2,3)	33	00:00:24	2.9	B.&M.	3.00	4.00	0.00	0.00
OA(32,9,2,3)	34	00:00:08	2.7	B.&M.	6.00	8.00	0.00	0.00
OA(32,10,2,3)	32	00:00:07	2.7	B.&M.	10.00	16.00	0.00	0.00
OA(32,11,2,3)	22	00:00:07	2.7	B. (2004)	25.00	0.00	27.00	0.00
OA(32,12,2,3)	23	00:00:05	2.7	B. (2004)	38.00	0.00	52.00	0.00
OA(32,13,2,3)	12	00:00:04	2.7	B. (2004)	55.00	0.00	96.00	0.00
OA(32,14,2,3)	10	00:00:03	2.7	B. (2003)*	77.00	0.00	168.00	0.00
OA(32,15,2,3)	5	00:00:02	2.7	B. (2003)*	105.00	0.00	280.00	0.00
OA(32,16,2,3)	4	00:00:02	2.7	B. (2003)*	140.00	0.00	448.00	0.00
OA(36,3,2,2'')	1	00:04:41	3.5	S.&T.	0.01			
OA(36,4,2,2'')	1	00:00:35	3.5	S.&T.	0.05	0.01		
OA(36,5,2,2'')	5	00:00:03	3.5	E.,K.,&L.	0.12	0.06	0.00	

Continued on next page

Table 5 – continued from previous page

$OA(N, k, s, t)$	Classes	Time	Algorithm	Status	$A_{t+1}$	$A_{t+2}$	$A_{t+3}$	$A_{t+4}$
OA(36,6,2,2'')	652	00:04:47	3.5	A., K., &L	0.25	0.19	0.00	0.44
OA(36,7,2,2'')	176929	82:59:45	3.5		0.43	0.43	1.04	0.35
OA(36,8,2,2'')	1320951	1797:40:43	3.5		0.69	1.75	2.77	0.79
OA(36,9,2,2'')	503	949:40:58	3.5		1.04	3.63	5.63	2.37
OA(36,10,2,2'')	0	00:07:36	3.5					
OA(40,5,2,3)	3	00:00:01	2.9	S.&T.	0.20	0.00		
OA(40,6,2,3)	9	00:00:03	2.9	B.&M.	0.60	0.00	0.00	
OA(40,7,2,3)	25	00:00:07	2.9	B.&M.	1.40	0.00	0.80	0.00
OA(40,8,2,3)	105	00:01:15	2.9	B.&M.	2.80	2.56	0.64	0.00
OA(40,9,2,3)	213	00:01:40	2.7	B.&M.	5.04	5.76	1.92	0.00
OA(40,10,2,3)	353	00:02:29	2.7	B.&M.	8.40	11.52	4.80	0.00
OA(40,11,2,3)	260	00:03:25	2.7		18.96	0.00	22.72	0.00
OA(40,12,2,3)	235	00:02:40	2.7		28.44	0.00	45.44	0.00
OA(40,13,2,3)	132	00:02:29	2.7		41.72	0.00	81.92	0.00
OA(40,14,2,3)	96	00:01:25	2.7		58.60	0.00	142.40	0.00
OA(40,15,2,3)	36	00:01:00	2.7		80.20	0.00	236.00	0.00
OA(40,16,2,3)	26	00:00:22	2.7		107.04	0.00	376.96	0.00
OA(40,17,2,3)	7	00:00:12	2.7		140.00	0.00	582.40	0.00
OA(40,18,2,3)	6	00:00:03	2.7		180.00	0.00	873.60	0.00
OA(40,19,2,3)	3	00:00:02	2.7		228.00	0.00	1276.80	0.00
OA(40,20,2,3)	3	00:00:01	2.7		285.00	0.00	1824.00	0.00
OA(48,4,2,3)	4	00:00:01	2.9	S.&T.	0.00			
OA(48,5,2,3)	10	00:00:01	2.9	S.&T.	0.00	0.11		
OA(48,6,2,3)	45	00:00:04	2.9	B.&M.	0.11	0.44	0.00	
OA(48,7,2,3)	397	00:01:37	2.9	B.&M.	0.33	1.33	0.00	0.00
OA(48,8,2,3)	8383	01:14:47	2.7	B.&M.	1.00	2.67	0.67	0.00
OA(48,9,2,3)	166081	23:45:26	2.7		2.44	5.33	1.78	0.00
OA(48,10,2,3)	1310006	251:18:53	2.7		5.33	8.67	4.00	1.33
OA(48,11,2,3)	3528089	976:02:53	2.7		9.11	14.22	9.78	3.56
OA(48,12,2,3)	4460865	1746:22:44	2.7		15.00	24.89	10.67	14.22
OA(48,13,2,3)	3980095	1986:17:11	2.7		23.00	32.00	37.33	21.33
OA(48,14,2,3)	3139653	1908:09:20	2.7		34.33	53.33	40.00	64.00
OA(48,15,2,3)	2165144	1309:29:37	2.7		64.11	0.00	204.44	0.00
OA(48,16,2,3)	1288460	963:52:42	2.7	B. (2004)*	85.67	0.00	326.00	0.00
OA(48,17,2,3)	629705	527:08:22	2.7		112.67	0.00	500.00	0.00
OA(48,18,2,3)	259346	260:40:43	2.7		145.33	0.00	746.67	0.00
OA(48,19,2,3)	84495	100:14:05	2.7		184.44	0.00	1088.89	0.00
OA(48,20,2,3)	24012	31:15:26	2.7	B. (2004)*	230.67	0.00	1554.67	0.00
OA(48,21,2,3)	4919	08:02:33	2.7	B. (2004)*	285.00	0.00	2176.00	0.00
OA(48,22,2,3)	1129	01:33:26	2.7	B. (2003)*	348.33	0.00	2992.00	0.00
OA(48,23,2,3)	130	00:26:07	2.7	B. (2003)*	421.67	0.00	4048.00	0.00
OA(48,24,2,3)	36	00:07:44	2.7	B. (2003)*	506.00	0.00	5397.33	0.00
OA(48,25,2,3)	0	00:00:34	2.7					
OA(81,4,3,3)	32	00:00:17	2.9		0.00			

Continued on next page

Table 5 – continued from previous page

$OA(N, k, s, t)$	Classes	Time	Algorithm	Status	$A_{t+1}$	$A_{t+2}$	$A_{t+3}$	$A_{t+4}$
OA(81,5,3,3)	17056	04:33:34	2.9		0.00	2.00		
OA(81,6,3,3)	1099	37:07:45	2.7		4.00	4.00	0.00	
OA(81,7,3,3)	486	01:32:30	2.7		10.00	12.00	2.00	2.00
OA(81,8,3,3)	235	00:15:38	2.5		20.00	32.00	8.00	16.00
OA(81,9,3,3)	1	00:00:13	2.5		36.00	72.00	24.00	72.00
OA(81,10,3,3)	1	00:00:02	2.5		60.00	144.00	60.00	240.00
OA(81,11,3,3)	0	00:00:01	2.5					
OA(160,7,2,4)	450	00:05:17	2.9		0.00	0.28	0.00	
OA(160,8,2,4)	99618	539:41:48	2.10		0.20	0.52	0.04	0.00
OA(160,9,2,4)	15083	11306:19:23	2.8		0.56	1.36	0.24	0.04
OA(160,10,2,4)	0	08:52:05	2.5					

Table 6: Distance distributions of GMA designs.

$OA(N, k, s, t)$	$B_0, B_1, \dots, B_k$
OA(24,3,2,2)	3, 9, 9, 3
OA(24,4,2,2)	1.667, 5.333, 10, 5.333, 1.667
OA(24,5,2,2)	1.167, 2.500, 8.333, 8.333, 2.500, 1.167
OA(24,6,2,2)	1, 1, 5, 10, 5, 1, 1
OA(24,7,2,2)	1, 0.167, 2.500, 8.333, 8.333, 2.500, 0.167, 1
OA(24,8,2,2)	1, 0, 0.667, 5.333, 10, 5.333, 0.667, 0, 1
OA(24,9,2,2)	1, 0, 0, 2, 9, 9, 2, 0, 0, 1
OA(24,10,2,2)	1, 0, 0, 0, 5, 12, 5, 0, 0, 0, 1
OA(24,11,2,2)	1, 0, 0, 0, 0, 11, 11, 0, 0, 0, 0, 1
OA(24,12,2,2)	1, 0, 0, 0, 0, 0, 22, 0, 0, 0, 0, 0, 1
OA(24,13,2,2)	1, 0, 0, 0, 0, 0, 10, 12, 0, 0, 0, 0, 1, 0
OA(24,14,2,2)	1, 0, 0, 0, 0, 0, 4, 12, 6, 0, 0, 0, 1, 0, 0
OA(24,15,2,2)	1, 0, 0, 0, 0, 0, 1.333, 8, 10, 2.667, 0, 0, 1, 0, 0, 0
OA(24,16,2,2)	1, 0, 0, 0, 0, 0, 0.333, 4, 10, 6.667, 1, 0, 1, 0, 0, 0, 0
OA(24,17,2,2)	1, 0, 0, 0, 0, 0, 0, 1.333, 7.667, 9.333, 2.667, 1.333, 0.667, 0, 0, 0, 0, 0
OA(24,18,2,2)	1, 0, 0, 0, 0, 0, 0, 0, 3, 12, 6, 0, 2, 0, 0, 0, 0, 0
OA(24,19,2,2)	1, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 3, 2, 0, 0, 0, 0, 0, 0
OA(24,20,2,2)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 18, 0, 5, 0, 0, 0, 0, 0, 0, 0
OA(24,21,2,2)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 12, 5, 0, 0, 0, 0, 0, 0, 0
OA(24,22,2,2)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12, 11, 0, 0, 0, 0, 0, 0, 0, 0
OA(24,23,2,2)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 0, 0, 0, 0, 0, 0, 0, 0
OA(28,3,2,2)	3.571, 10.286, 10.714, 3.429
OA(28,4,2,2)	1.929, 6.571, 10.714, 7.143, 1.643
OA(28,5,2,2)	1.143, 3.929, 8.571, 9.286, 4.286, 0.786
OA(28,6,2,2)	1.071, 0.429, 10.714, 2.857, 11.786, 0.429, 0.714
OA(28,7,2,2)	1, 0, 4.786, 8.857, 6.714, 4.857, 1.786, 0
OA(28,8,2,2)	1, 0, 1.429, 7.429, 9, 5.143, 2.857, 1.143, 0
OA(28,9,2,2)	1, 0, 0, 4.500, 9.643, 6.429, 3.857, 1.929, 0.643, 0
OA(28,10,2,2)	1, 0, 0, 1.143, 7.429, 10, 4.429, 2.286, 1.143, 0.571, 0

Continued on next page

Table 6 – continued from previous page

$OA(N, k, s, t)$	$B_0, B_1, \dots, B_k$
OA(28,11,2,2)	1, 0, 0, 0.214, 3.643, 9.429, 7.929, 4, 0.929, 0, 0.786, 0.071
OA(28,12,2,2)	1, 0, 0, 0, 0.643, 7.714, 9.857, 5.143, 2.786, 0, 0, 0.857, 0
OA(28,13,2,2)	1, 0.071, 0, 0, 0, 2.571, 9.143, 11.286, 3, 0, 0, 0, 0.857, 0.071
OA(28,14,2,2)	1, 0.071, 0, 0, 0, 0, 5.571, 13, 7.429, 0, 0, 0, 0, 0.929, 0
OA(32,5,2,3)	1, 5, 10, 10, 5, 1
OA(32,6,2,3)	1, 0, 15, 0, 15, 0, 1
OA(32,7,2,3)	1, 0, 5, 12, 7, 4, 3, 0
OA(32,8,2,3)	1, 0, 1, 10, 11, 4, 3, 2, 0
OA(32,9,2,3)	1, 0, 0, 4, 14, 8, 0, 4, 1, 0
OA(32,10,2,3)	1, 0, 0, 0, 10, 16, 0, 0, 5, 0, 0
OA(32,11,2,3)	1, 0, 0, 0, 5, 10, 10, 5, 0, 0, 0, 1
OA(32,12,2,3)	1, 0, 0, 0, 1, 8, 12, 8, 1, 0, 0, 0, 1
OA(32,13,2,3)	1, 0, 0, 0, 0, 3, 12, 12, 3, 0, 0, 0, 1
OA(32,14,2,3)	1, 0, 0, 0, 0, 0, 7, 16, 7, 0, 0, 0, 0, 1
OA(32,15,2,3)	1, 0, 0, 0, 0, 0, 0, 15, 15, 0, 0, 0, 0, 0, 1
OA(32,16,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 30, 0, 0, 0, 0, 0, 0, 1
OA(36,3,2,2)	4.556, 13.333, 13.667, 4.444
OA(36,4,2,2)	2.389, 8.667, 13.667, 9.111, 2.167
OA(36,5,2,2)	1.333, 5.278, 11.111, 11.667, 5.556, 1.056
OA(36,6,2,2)	1.056, 1.667, 11.667, 6.667, 12.500, 1.667, 0.778
OA(36,7,2,2)	1, 0, 8.167, 7.778, 11.667, 4.667, 2.722, 0
OA(36,8,2,2)	1, 0, 2.667, 9.778, 10.333, 6.667, 4.222, 1.333, 0
OA(36,9,2,2)	1, 0, 0.333, 7, 11, 8, 5, 3, 0.667, 0
OA(40,5,2,3)	1.500, 5.500, 13, 13, 5.500, 1.500
OA(40,6,2,3)	1, 3, 9, 14, 9, 3, 1
OA(40,7,2,3)	1, 0.500, 7.500, 11, 11, 7.500, 0.500, 1
OA(40,8,2,3)	1.100, 0, 2.400, 11.200, 13, 6.400, 4, 1.600, 0.300
OA(40,9,2,3)	1.100, 0, 0, 7.200, 14.400, 9, 3.600, 3.600, 0.900, 0.200
OA(40,10,2,3)	1.100, 0, 0, 0, 18, 7.200, 9, 0, 4.500, 0, 0.200
OA(40,11,2,3)	1, 0, 0, 1.200, 6.400, 11.400, 11.400, 6.400, 1.200, 0, 0, 1
OA(40,12,2,3)	1, 0, 0, 0, 3.600, 9.600, 11.600, 9.600, 3.600, 0, 0, 0, 1
OA(40,13,2,3)	1, 0, 0, 0, 0.900, 6.300, 11.800, 11.800, 6.300, 0.900, 0, 0, 0, 1
OA(40,14,2,3)	1, 0, 0, 0, 0, 3, 9, 14, 9, 3, 0, 0, 0, 0, 1
OA(40,15,2,3)	1, 0, 0, 0, 0, 0.500, 5.500, 13, 13, 5.500, 0.500, 0, 0, 0, 0, 1
OA(40,16,2,3)	1, 0, 0, 0, 0, 0, 1.600, 9.600, 15.600, 9.600, 1.600, 0, 0, 0, 0, 0, 1
OA(40,17,2,3)	1, 0, 0, 0, 0, 0, 0, 4, 15, 15, 4, 0, 0, 0, 0, 0, 1
OA(40,18,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 9, 20, 9, 0, 0, 0, 0, 0, 0, 1
OA(40,19,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 0, 19, 19, 0, 0, 0, 0, 0, 0, 0, 1
OA(40,20,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 38, 0, 0, 0, 0, 0, 0, 0, 0, 1
OA(48,4,2,3)	3, 12, 18, 12, 3
OA(48,5,2,3)	1.667, 6.667, 16.667, 13.333, 8.333, 1.333
OA(48,6,2,3)	1.167, 3, 12.833, 15.333, 9.500, 5.667, 0.500
OA(48,7,2,3)	1, 1, 8, 16, 11, 7, 4, 0
OA(48,8,2,3)	1, 0, 4, 14, 13, 8, 6, 2, 0

Continued on next page

Table 6 – continued from previous page

$OA(N, k, s, t)$	$B_0, B_1, \dots, B_k$
OA(48,9,2,3)	1, 0, 0.667, 10, 15.333, 9.333, 6, 4.667, 1, 0
OA(48,10,2,3)	1, 0, 0, 4, 15, 13, 6, 6, 2, 1, 0
OA(48,11,2,3)	1, 0, 0, 0.333, 10.333, 17.667, 6.333, 5.667, 5.333, 0.333, 1, 0
OA(48,12,2,3)	1, 0, 0, 0, 4, 13.333, 17.333, 5.333, 1.667, 2.667, 2.667, 0, 0
OA(48,13,2,3)	1, 0, 0, 0, 0, 9, 22, 6, 0, 9, 0, 0, 1, 0
OA(48,14,2,3)	1, 0, 0, 0, 1, 0, 15, 24, 0, 0, 5, 0, 2, 0, 0
OA(48,15,2,3)	1, 0, 0, 0, 0, 1.667, 9, 12.333, 12.333, 9, 1.667, 0, 0, 0, 0, 1
OA(48,16,2,3)	1, 0, 0, 0, 0, 0, 5, 12, 12, 12, 5, 0, 0, 0, 0, 1
OA(48,17,2,3)	1, 0, 0, 0, 0, 0, 1, 9, 13, 13, 9, 1, 0, 0, 0, 0, 1
OA(48,18,2,3)	1, 0, 0, 0, 0, 0, 0, 4, 11, 16, 11, 4, 0, 0, 0, 0, 0, 1
OA(48,19,2,3)	1, 0, 0, 0, 0, 0, 0, 0.667, 7, 15.333, 15.333, 7, 0.667, 0, 0, 0, 0, 0, 0, 1
OA(48,20,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 2, 12, 18, 12, 2, 0, 0, 0, 0, 0, 0, 1
OA(48,21,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 0, 5, 18, 18, 5, 0, 0, 0, 0, 0, 0, 1
OA(48,22,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 0, 11, 24, 11, 0, 0, 0, 0, 0, 0, 0, 1
OA(48,23,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 23, 0, 0, 0, 0, 0, 0, 0, 0, 1
OA(48,24,2,3)	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 46, 0, 0, 0, 0, 0, 0, 0, 0, 1
OA(81,4,3,3)	1, 8, 24, 32, 16
OA(81,5,3,3)	1, 0, 20, 20, 30, 10
OA(81,6,3,3)	1, 0, 4, 24, 24, 20, 8
OA(81,7,3,3)	1, 0, 0, 10, 30, 18, 16, 6
OA(81,8,3,3)	1, 0, 0, 0, 20, 32, 8, 16, 4
OA(81,9,3,3)	1, 0, 0, 0, 0, 36, 24, 0, 18, 2
OA(81,10,3,3)	1, 0, 0, 0, 0, 0, 60, 0, 0, 20, 0
OA(160,7,2,4)	1.600, 7, 29.400, 42, 42, 29.400, 7, 1.600
OA(160,8,2,4)	1.100, 3.300, 18.900, 36.700, 40.500, 35.900, 18.700, 4.100, 0.800
OA(160,9,2,4)	1, 0.900, 11.400, 30, 37, 37, 28.600, 11.600, 2, 0.500

Table 2: Extending  $OA(20, k-1, 2, 2)$  to  $OA(20, k, 2, 2)$  with real times (minutes:seconds).

OA(20, k, 2, 2)		Algorithm 2.4		Algorithm 2.5		Algorithm 2.7		Algorithm 2.8		Algorithm 2.10	
$k$	$ T_k $	$ M $	Time	$ M $	Time	$ M $	Time	$ M $	Time	$ M $	Time
3	3	21252	0:45.06	5	0:00.10	5	0:00.16	5	0:00.15	3	0:01.00
4	3	13824	0:40.95	44	0:00.38	16	0:00.36	16	0:00.36	4	0:00.52
5	11	6588	0:24.52	363	0:02.06	44	0:00.46	44	0:00.48	27	0:00.81
6	75	7740	0:35.94	1659	0:09.30	661	0:03.43	661	0:03.61	302	0:08.70
7	474	16526	1:36.89	5679	0:38.37	4273	0:26.29	4273	0:26.03	2911	3:11.39
8	1603	33272	4:39.94	15219	2:14.77	14080	2:05.29	14080	2:35.87	12202	50:51.42
9	2477	46028	8:18.19	22744	4:11.31	21883	4:59.71	21883	5:34.78	21051	474:12.00
10	2389	48148	9:10.39	23984	4:56.12	23144	6:37.18	23144	6:58.68		
11	1914	42298	9:53.56	21149	5:33.74	20452	7:30.26	20452	8:02.59		
12	1300	30544	7:49.23	15272	4:19.80	14822	5:47.66	14822	6:18.30		
13	730	18200	5:01.89	9100	2:48.59	8797	3:57.74	8797	3:52.29		
14	328	8760	2:44.05	4380	1:31.58	4213	2:13.20	4213	2:08.32		
15	124	3280	1:06.88	1640	0:38.18	1560	0:54.57	1560	0:54.18		
16	40	992	0:22.20	496	0:12.26	455	0:19.02	455	0:18.91		
17	11	240	0:05.70	120	0:03.38	106	0:05.59	106	0:05.65		
18	6	44	0:00.98	22	0:00.64	19	0:01.44	19	0:01.25		
19	3	12	0:00.36	6	0:00.27	6	0:00.64	6	0:00.65		

Table 3: Extending  $OA(20, k-1, 2, 2')$  to  $OA(20, k, 2, 2')$  with real times in seconds using Algorithm 3.2.

$k$	$ T'_k $	$ M' $	Time
3	1	4	0.16
4	2	14	0.19
5	4	24	0.37
6	13	236	1.84
7	21	262	4.52
8	6	40	4.06
9	2	6	0.92
10	1	4	0.37
11	0	0	0.17

Table 4: Extending  $OA(20, k-1, 2, 2'')$  to  $OA(20, k, 2, 2'')$  defined by  $A_4 \leq 13.75$  when  $k_{\text{input}} = 10$  with real times in seconds using Algorithm 3.5.

$k$	$ T''_k $	$ M'' $	Time
3	1	4	0.18
4	1	8	0.19
5	2	8	0.22
6	2	14	1.10
7	2	12	0.49
8	0	0	0.32

Table 7: Number of input  $OA(N, k_0, s, t)$  tested with Algorithm 5.1. An asterisk indicates that a set of all non-isomorphic saturated designs was tested.

$(N, k, s)$	OA( $N, N-1, 2, 2$ )		OA( $N, k_0, s, t$ )	Inputs	Total
	$H_N$	Inputs			
(24, k, 2)	$H_{24}$	*130			130
(28, k, 2)	$H_{28}$	*7,570			7,570
(32, k, 2)	$H_{16} \otimes H_2$	5			5
(36, k, 2)	$H_{36}$	2			2
(40, k, 2)	$H_{20} \otimes H_2$	3	OA(40,20,2,3)	1	4
(48, k, 2)	$H_{24} \otimes H_2$	130	OA(48,24,2,3)	1	131
(81, k, 3)			OA(81,40,3,2)	1	1
(160, k, 2)	$H_{20} \otimes H_8$	2	OA(160,80,2,3)	1	3

Table 8: The subset of  $OA(N, k, s, t)$  from Table 6 where Algorithm 5.1 did not find a known GMA design, the number of isomorphism classes of GMA  $OA(N, k, s, t)$ , and whether or not a GMA  $OA(N, k, s, t)$  extends to a larger design (determined by repeated application of basic Algorithm 2.1). (Unknown cases, indicated by question marks, were out of our computational reach.)

$OA(N, k, s, t)$	GMA design(s)	Larger design(s)	Extends
$OA(28, 6, 2, 2)$	1	$OA(28, 27, 2, 2)$	?
$OA(28, 10, 2, 2)$	2	$OA(28, 27, 2, 2)$	no
$OA(28, 11, 2, 2)$	6	$OA(28, 27, 2, 2)$	no
$OA(28, 12, 2, 2)$	2	$OA(28, 27, 2, 2)$	no
$OA(36, 7, 2, 2)$	1	$OA(36, 35, 2, 2)$	?
$OA(36, 8, 2, 2)$	21,562	$OA(36, 35, 2, 2)$	?
$OA(36, 9, 2, 2)$	503	$OA(36, 35, 2, 2)$	?
$OA(40, k, 2, 3)$	1 for each $k = 7, 8, 9, 10$	$OA(40, 20, 2, 3), OA(40, 39, 2, 2)$	no, ?
$OA(48, 13, 2, 3)$	1	$OA(48, 24, 2, 3), OA(48, 47, 2, 2)$	no, ?
$OA(160, k, 2, 4)$	1 for each $k = 7, 8, 9$	$OA(160, 80, 2, 3), OA(160, 159, 2, 2)$	?, ?

Table 9: Algorithm 5.2 was run with the listed  $(N, s, t)$  and ILP (altered to an objective function with randomly selected binary coefficients). Outputs are the subset of  $(N, k, s, t)$  listed in Table 6 where a known GMA design was found,  $k_{\max}$ , and approximate times (hours:minutes) and values “current” when an  $OA(N, k, s, t)$  was last overwritten after finding a better design.

$(N, s, t)$	Inputs		Outputs			
	ILP	$k$ with GMA	$OA(N, k, s, t)$	$k_{\max}$	Time	Current
(24, 2, 2)	Algorithm 2.7		3-23	23	12:00	60,000
(28, 2, 2)	Algorithm 3.2		3-5, 8, 10-14	14	3:00	30,000
(32, 2, 3)	Algorithm 2.7		5-16	16	0:10	3,000
(36, 2, 3)	Algorithm 3.2		3-5	16	18:00	70,000
(40, 2, 3)	Algorithm 2.7		5-20	20	5:00	30,000
(48, 2, 3)	Algorithm 2.7		6, 7, 12, 15-24	24	36:00	50,000
(81, 3, 3)	Algorithm 2.7		5-7	8	0:01	50
(160, 2, 4)	Algorithm 2.7		7	9	1:00	80

Table 10: Partial GWPs of weak GMA or near GMA  $OA(36, k, 2, 2)$  with  $k > 6$  obtained via Algorithms 5.1 and 5.2.

$OA(N, k, s, t)$	Algorithm	$A_3$	$A_4$	$A_5$	$A_6$
$OA(36, 7, 2, 2)$	5.1	0.43	0.73	0.94	0.44
$OA(36, 8, 2, 2)$	5.2	0.69	1.85	2.57	0.89
$OA(36, 9, 2, 2)$	5.2	1.04	3.93	5.38	2.07
$OA(36, 10, 2, 2)$	5.2	1.48	7.33	9.93	4.69
$OA(36, 11, 2, 2)$	5.2	2.04	13.46	13.33	12.54
$OA(36, 12, 2, 2)$	5.2	2.72	22.70	16.79	28.54
$OA(36, 13, 2, 2)$	5.2	3.53	33.91	24.44	54.72
$OA(36, 14, 2, 2)$	5.2	4.49	47.81	37.78	94.72
$OA(36, 15, 2, 2)$	5.2	5.62	65.44	56.64	156.79
$OA(36, 16, 2, 2)$	5.2	6.91	87.26	82.57	250.86
$OA(36, 17, 2, 2)$	5.1	8.40	123.41	100.94	423.11
$OA(36, 18, 2, 2)$	5.1	10.07	158.67	141.04	634.67
$OA(36, 19, 2, 2)$	5.1	17.30	168.74	261.33	775.70

Table 11: Partial GWPs of near GMA  $OA(28, k, 2, 2)$  with  $k > 14$  obtained via Algorithms 5.1 and 5.2.

$OA(N, k, s, t)$	Algorithm	$A_3$	$A_4$	$A_5$	$A_6$
$OA(28, 15, 2, 2)$	5.2	12.71	72.43	97.71	156.00
$OA(28, 16, 2, 2)$	5.2	17.31	87.43	145.63	263.18
$OA(28, 17, 2, 2)$	5.2	23.18	105.06	218.20	407.84
$OA(28, 18, 2, 2)$	5.2	29.39	129.06	306.12	623.02
$OA(28, 19, 2, 2)$	5.2	36.59	159.10	412.73	924.90
$OA(28, 20, 2, 2)$	5.2	43.84	196.67	549.22	1330.29
$OA(28, 21, 2, 2)$	5.2	51.63	241.82	718.04	1870.37
$OA(28, 22, 2, 2)$	5.1, 5.2	60.82	293.45	922.20	2588.41
$OA(28, 23, 2, 2)$	5.1, 5.2	70.43	354.59	1174.12	3507.67
$OA(28, 24, 2, 2)$	5.1, 5.2	80.49	425.51	1483.10	4676.90
$OA(28, 25, 2, 2)$	5.1, 5.2	92.00	506.00	1848.00	6160.00
$OA(28, 26, 2, 2)$	5.1, 5.2	104.00	598.00	2288.00	8008.00
$OA(28, 27, 2, 2)$	$H_{28}$	117.00	702.00	2808.00	10296.00