Symmetry Groups for Linear Programming Relaxations of Orthogonal Array Problems

THESIS

MARCH 2015

David M. Arquette, Second Lieutenant, USAF

AFIT-ENC-MS-15-M-003

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENC-MS-15-M-003

SYMMETRY GROUPS FOR LINEAR PROGRAMMING RELAXATIONS OF ORTHOGONAL ARRAY PROBLEMS

THESIS

Presented to the Faculty

Department of Mathematics and Statistics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Applied Mathematics

David M. Arquette, B.S.

Second Lieutenant, USAF

MARCH 2015

AFIT-ENC-MS-15-M-003

SYMMETRY GROUPS FOR LINEAR PROGRAMMING RELAXATIONS OF ORTHOGONAL ARRAY PROBLEMS

David M. Arquette, B.S.
Second Lieutenant, USAF

AFIT-ENC-MS-15-M-003

Abstract

Integer linear programs arise in many situations, and solving such problems can be computationally demanding. One way to solve them more efficiently is by exploiting the symmetry within their formulation. This paper proves that the symmetry group for the linear programming relaxation of 2-level orthogonal array problems of strength 2 is a particular semidirect product.

I dedicate this thesis to my wife, mother, and father for their unconditional love and support.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Dursun A. Bulutoglu, without whom this thesis would not have been possible. Thank you for matching my skills with an appropriate problem and for all of the time you set aside to help me with my research.

David M. Arquette

Table of Contents

SYMMETRY GROUPS FOR LINEAR PROGRAMMING RELAXATIONS OF ORTHOGONAL ARRAY PROBLEMS

I. INTRODUCTION

1.1 Motivation

Solving integer linear programs (ILPs) is a common problem, and it is not always an easy task. In fact, doing so often requires a substantial amount of compuational time. There are several methods available to solve ILPs more efficiently. In particular, the Margot ILP solver is able to decrease computation time by orders of magnitude by taking advantage of symmetry within the formulation of a problem [4]. In order to take full advantage of the symmetry in a given formulation, one must know as much about the symmetry group as possible. Ideally, the entire symmetry group will be known. Geyer [2] developed an algorithm for finding the symmetry group for linear programming (LP) relaxations of ILPs with equality constraints by using projection matrices. The goal of this research is to prove a special case of a conjecture resulting from Geyer [2].

1.2 Research Contribution

This research has proven that the symmetry group for the LP relaxation of an ILP formulation of a 2-level, k-factor, strength 2 orthogonal array is $S_2^k \rtimes S_{k+1}$. At face value, this result provides the symmetry group for the LP relaxation of certain ILPs. The methods used to prove this result may also be useful for finding other such symmetry groups with different strengths and levels. Furthermore, this result provides a theoretical verification of Geyer's [2] algorithm. Finally, the methods employed in this research may be useful for developing more efficient computational algorithms.

1.3 Organization of Thesis

This thesis is divided into four chapters and an appendix. Chapter 2 contains a brief discussion of some literature that is directly related to this research. Chapter 3 covers the original work performed and is divided into three sections. The first lays the groundwork for the work that follows. The second section walks through the relatively simple case of a strength one orthogonal array, and the third delves into the strength two case. Chapter 4 outlines some topics for future research that relate to this work. The appendix contains the code that was used to validate one of the proofs.

II. LITERATURE REVIEW

Experiments are generally intended to allow insight into situations. How an experiment is designed is of great importance to the effectiveness of the experiment and the relevance of the results. A factorial design is a collection of factors that assume a finite number of level combinations. If one design can be obtained from another by permuting runs, factors, or levels, the two designs are said to be *isomorphic.* For certain linear models, orthogonal arrays are the most efficient class of factorial designs. An orthogonal array with N runs (rows), k factors (columns), s levels, and strength t is denoted $\mathrm{OA}(N, k, s, t)$. The index of such an orthogonal array, λ , is the number of times every t-tuple appears within each combination of t columns. The index is typically omitted from notation because $\lambda = N/s^t$.

If an ILP contains any variables that can be permuted without changing the feasibility and optimality of its solutions, it is said to be *symmetric.* Margot [4] defined the *symmetry group*, G , of an ILP to be

$$G = \{\pi \in S_n | \mathbf{c}^T\mathbf{x} = \mathbf{c}^T\pi(\mathbf{x}) \text{ and } \pi(\mathbf{x}) \in \mathcal{F} \; \forall \mathbf{x} \in \mathcal{F}\}$$

where $\mathbf{c}^T\mathbf{x}$ is the objective function of the ILP, and $\mathcal{F}$ is the set of all feasible solutions. Symmetric ILPs can arise from a variety of problem formulations. In particular, ILPs for enumerating orthogonal arrays are highly symmetric.

Optimal solutions to ILPs are commonly found with branch-and-bound or branch-and-cut algorithms. In the case of symmetric ILPs, many of the subproblems in the enumeration tree are isomorphic. As a result, a considerable amount of computational time is wasted on solving identical problems repeatedly. Thankfully, Margot [4] developed a solver that is able to decrease and potentially eliminate such redundant computations by exploiting a subgroup of an ILP's symmetry group when pruning its enumeration tree. Exploiting larger subgroups results in increased reductions in redundant computations, and the greatest reduction is attained if the full symmetry group of the ILP is exploited. Hence, it is desirable to find larger subgroups of the symmetry group of an ILP.

For an ILP with objective function $\mathbf{c}^T\mathbf{x}$, $m \times n$ constraint matrix $\mathbf{A}$, and righthand side $\mathbf{b}$ with $\mathbf{0} \leq \mathbf{x} \leq \mathbf{d}$, let $\mathbf{A}(\pi, \sigma)$ be the matrix found by permuting the rows and columns of $\mathbf{A}$ according σ and π , respectively. The *automorphism group* of such an ILP is

$$G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d}) = \{\pi \in S_n | \pi(\mathbf{c}) = \mathbf{c}, \pi(\mathbf{d}) = \mathbf{d}, \text{ and } \exists \sigma \in S_m : \mathbf{A}(\pi, \sigma) = \mathbf{A}, \sigma(\mathbf{b}) = \mathbf{b}\}$$

where σ preserves equalities and inequalities. Clearly, $G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d}) \leq G$, and computational experiments suggest that efficiency can be improved by several orders of magnitude if Margot's solver is used with $G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ on highly symmetric ILPs [4].

Because the makeup of the symmetry group, G , is determined by the ILP's feasible set, identifying all symmetries in an ILP is quite difficult and remains an open problem. Margot [4] proved that deciding if $G = S_n$ is *NP-Complete.* Therefore, finding G for any given ILP is *NP-Hard.* In order to find many of the symmetries in an ILP, one can simply find the symmetry group of the LP relaxation of the ILP. The LP relaxation symmetry group, G^{LP} is the set of all permutations of variables that send LP feasible points to LP feasible points with the same objective function value. For an LP relaxation without equality constraints where each constraint in $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ is a facet (non-redundant), $G^{LP} = G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d})$. Let $\mathcal{F}(k, s, t)$ and $\mathcal{F}(k, s, t)^{LP}$ be the sets of feasible solutions of an ILP and its LP relaxation, respectfully. Note that $\mathcal{F}(k, s, t) \subseteq \mathcal{F}(k, s, t)^{LP}$, so $G^{LP} \leq G$.

For an LP relaxation with equality constraints, $G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d}) \subseteq G^{LP}$. Geyer [2] developed an algorithm for finding G^{LP} of such an LP. Furthermore, Geyer [2] observed that $G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d}) \subset G^{LP}$ for the LP relaxation of an ILP formulation with equality constraints for finding $\mathrm{OA}(N, k, 2, t)$ when t is even. Geyer, Bulutoglu, and Rosenberg [3] explicitly found $G(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{d})$ for this formulation. They also found G^{LP} for a formulation without equality constraints. This thesis verifies Geyer's computational observations by explicitly finding G^{LP} for an ILP formulation with equality constraints. The equality constraints of this ILP formulation are linear combinations of those of the ILP formulation with equalities in [3]. Furthermore, both ILPs have the same number of non-redundant equality constraints. Hence, one can go back and forth between the two ILP formulations by applying a sequence of row operations to the equality constraints of each. This implies that the feasible sets of the LP relaxations of these ILP formulations are the same, so G^{LP} must also be the same for the two ILPs.

The ILP formulation used in this thesis stems from the concept of J-characteristics. Let the frequency vector, $\mathbf{f}$, of a 2-level factorial design, $\mathbf{D}$, have the frequency of each of the 2^k possible factor level combinations as its entries. Hence, $\mathbf{f}$ determines $\mathbf{D}$ up to reordering of factor level combinations. For a 2-level design, $\mathbf{D}$, with N runs (factor level combinations) and k factors, *J-characteristics* are given by

$$J_l = \sum_{i=1}^{N} \prod_{j \in l} d_{ij}$$

for $l \subseteq \mathbb{Z}_k$. It has been shown that $\mathbf{D}$ is uniquely determined by its J-characteristics up to reordering of its runs; furthermore, $\mathbf{D}$ is an orthogonal array of strength t if and only if $J_l = 0$ for all $l \subseteq \mathbb{Z}_k$ with $|l| \leq t$, where $l \neq \emptyset$ [7].

III. ORIGINAL WORK

3.1 Preliminary Steps

Let $\mathbf{1}$ be the column vector of length 2^k for which every entry is one. For $i = 1, \ldots, k$, let $\mathbf{x}_i$ be the i^{th} column of the k-factor, 2-level (± 1) full factorial design. For distinct $i_1, \ldots, i_j \in \{1, \ldots, k\}$, let $\mathbf{x}_{i_1,\ldots,i_j}$ represent the j-factor interaction term given by the Hadamard product $\mathbf{x}_{i_1} \circ \cdots \circ \mathbf{x}_{i_j}$.

Consider the equation

$$\mathbf{Mf} = \mathbf{J} \tag{1}$$

from [7], where $\mathbf{M}$ is the $\sum\limits_{i=0}^{t} \binom{k}{i}$ by 2^k matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{1}^T \\ \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_k^T \\ \mathbf{x}_{1,2}^T \\ \vdots \\ \mathbf{x}_{k-t+1,\ldots,k}^T \end{bmatrix}$$

$\mathbf{J}$ is the quasi-lexicographically ordered J-characteristic vector with entries $\mathbf{J}_l$ for $|l| \leq t$, and $\mathbf{f}$ is the frequency vector of a hypothetical OA$(N, k, 2, t)$. Then by the result in [7]

$$\mathbf{J} = \begin{bmatrix} N \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Our goal is to find the subgroup of the permutation group S_{2^k} that sends feasible solutions ($\mathbf{f} \in \mathbb{Q}^{2^k}_{\geq 0}$) to feasible solutions. This is the symmetry group of the LP relaxation of the ILP based on Equation 1, where the objective function is taken to be the zero vector. The equality constraints of this ILP are linear combinations of those of the orthogonal array defining ILP in [3]. Both ILPs have the same inequality constraints, and each ILP has $\sum_{i=0}^{t} \binom{k}{i}$ non-redundant equality constraints. Hence, both ILPs have the same LP relaxation feasible set, and this implies that both have the same LP relaxation symmetry group. From this point on, we shall refer to this group as G .

Theorem 1 *The symmetry group G is precisely the intersection of the automorphism group of the row space of $\mathbf{M}$ and the permutation group S_{2^k} , written $Aut(Row(\mathbf{M})) \cap S_{2^k}$. That is, G is the set of permutations that preserve $Row(\mathbf{M})$.*

Proof Observe that

$$\mathbf{f}^* = \begin{bmatrix} \frac{N}{2^k} \\ \vdots \\ \frac{N}{2^k} \end{bmatrix}$$

is a particular solution to Equation 1. As such, every solution $\mathbf{f}$ can be written in the form $\mathbf{f}^* + \mathbf{f}'$ where $\mathbf{f}' \in Null(\mathbf{M})$. Let $g \in G$ be arbitrary. Then $g(\mathbf{f})$ is a solution to Equation 1. That is,

$$\mathbf{M}g(\mathbf{f}) = \mathbf{M}g(\mathbf{f}^* + \mathbf{f}') = \mathbf{J}$$

Because $g \in G \leq S_{2^k}$,

$$\mathbf{M}[g(\mathbf{f}^*) + g(\mathbf{f}')] = \mathbf{M}[\mathbf{f}^* + g(\mathbf{f}')] = \mathbf{J}$$

and thus

$$\mathbf{M}\mathbf{f}^* + \mathbf{M}g(\mathbf{f}') = \mathbf{J} + \mathbf{M}g(\mathbf{f}') = \mathbf{J}$$

Therefore,

$$\mathbf{M}g(\mathbf{f}') = \mathbf{0}$$

so we see that $g(\mathbf{f}') \in Null(\mathbf{M})$ which means g must preserve $Null(\mathbf{M})$. Because $g \in G$ was arbitrary, we have shown $G \leq Aut(Null(\mathbf{M})) \cap S_{2^k}$.

Now let $h \in Aut(Null(\mathbf{M})) \cap S_{2^k}$ be arbitrary. Then

$$\begin{aligned}
\mathbf{M}h(\mathbf{f}) &= \mathbf{M}h(\mathbf{f}^* + \mathbf{f}') \\
&= \mathbf{M}[h(\mathbf{f}^*) + h(\mathbf{f}')] \\
&= \mathbf{M}[\mathbf{f}^* + h(\mathbf{f}')] \\
&= \mathbf{M}\mathbf{f}^* + \mathbf{M}h(\mathbf{f}') \\
&= \mathbf{J} + \mathbf{0} \\
&= \mathbf{J}
\end{aligned}$$

Hence, $h \in G$, and because h was arbitrary, $Aut(Null(\mathbf{M})) \cap S_{2^k} \leq G$. Noting that $Aut(Null(\mathbf{M})) = Aut(Row(\mathbf{M}))$, we conclude that $G = Aut(Null(\mathbf{M})) \cap S_{2^k} = Aut(Row(\mathbf{M})) \cap S_{2^k}$.

3.2 The Symmetry Group for the Strength One Case

Before investigating the symmetry group for a strength two orthogonal array, it is only natural that we should address the strength one case. As such, for now we assume an OA$(N, k, 2, 1)$ in Equation 1. Let

$$B = \{\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_k\}$$

Clearly, B is an orthogonal basis for $Row(\mathbf{M})$. For all $g \in G$, we know $g(B)$ must also be an orthogonal basis for $Row(\mathbf{M})$ because by Theorem 1, $g \in Aut(Row(\mathbf{M})) \cap S_{2^k}$, and elements of S_{2^k} preserve angles. Furthermore, for every $\mathbf{x} \in B$, we know that $g(\mathbf{x})$ can be represented uniquely as a linear combination of the elements of B . That is,

$$g(\mathbf{x}) = \lambda_0 \mathbf{1} + \lambda_1 \mathbf{x}_1 + \cdots + \lambda_k \mathbf{x}_k \tag{2}$$

Lemma 1 *Let $\mathbf{x} \in B$. If $\mathbf{x} = \mathbf{1}$ in Equation 2, then $\lambda_0 = 1$, and $\lambda_i = 0$ for $i = 1, \ldots, k$. Otherwise, $\lambda_0 = 0$.*

Proof Suppose $\mathbf{x} = \mathbf{1}$. Because $g \in G \leq S_{2^k}$, $g(\mathbf{1}) = \mathbf{1}$ which uniquely satisfies Equation 2. For $i = 1, \ldots, k$, we know $g(\mathbf{x}_i)$ must be orthogonal to $g(\mathbf{1}) = \mathbf{1}$, so $\lambda_0 = 0$ whenever $\mathbf{x} \neq \mathbf{1}$.

Lemma 2 *If $\{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ is obtained from $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, the columns of the full factorial 2^k design, by permuting rows, then there exists a permutation $\sigma \in S_k$ such that for all $i \in \{1, \ldots, k\}$ satisfying $\mathbf{x}'_i \in span(\mathbf{x}_1, \ldots, \mathbf{x}_k)$, $\mathbf{x}'_i = \pm \mathbf{x}_{\sigma(i)}$.*

Proof Let $\{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ is obtained from $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, the columns of the full factorial 2^k design, by permuting rows. Suppose $\mathbf{x}'_i \in span(\mathbf{x}_1, \ldots, \mathbf{x}_k)$. Then $\mathbf{x}'_i = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_k \mathbf{x}_k$, and we have the system of equations

$$\begin{array}{cccccc}
\lambda_1 & + & \cdots & +\lambda_k & = & \pm 1 \\
\lambda_1 & + & \cdots & -\lambda_k & = & \pm 1 \\
\vdots & & \ddots & \vdots & & \vdots \\
-\lambda_1 & - & \cdots & -\lambda_k & = & \pm 1
\end{array}$$

Subtracting the second equation from the first equation gives $\lambda_k \in \{0, \pm 1\}$. Choosing other pairs of equations similarly yields $\lambda_j \in \{0, \pm 1\}$ for $j = 1, \ldots, k$. Because B is an orthogonal set, the Pythagorean Theorem gives

$$\begin{aligned}\|\mathbf{x}_i'\|^2 &= \sum_{j=1}^{k} \|\lambda_j \mathbf{x}_j\|^2 \\ &= \sum_{j=1}^{k} (|\lambda_j|\|\mathbf{x}_j\|)^2 \\ &= \sum_{j=1}^{k} |\lambda_j|^2 \|\mathbf{x}_j\|^2 \\ &= \sum_{j=1}^{k} \lambda_j^2 \|\mathbf{x}_j\|^2\end{aligned}$$

Because row permutations are norm-preserving, $\|\mathbf{x}_i'\| = 2^k = \|\mathbf{x}_j\|$ for $j = 1, \ldots, k$. Thus,

$$\sum_{j=1}^{k} \lambda_j^2 = 1$$

Because $\lambda_j \leq \{0, \pm 1\}$ for $j = 1, \ldots, k$, there is exactly one nonzero $\lambda_j \in \{\pm 1\}$, and $\mathbf{x}_i' = \pm \mathbf{x}_j$. Row permutations also preserve orthogonality, so for every distinct $i \in \{1, \ldots, k\}$ such that $\mathbf{x}_i' \in span(\mathbf{x}_1, \ldots, \mathbf{x}_k)$, there is a unique $j \in \{1, \ldots, k\}$ satisfying $\mathbf{x}_i' = \pm \mathbf{x}_j$. Thus, there exists a permutation $\sigma \in S_k$ such that $\mathbf{x}_i' = \pm \mathbf{x}_{\sigma(i)}$ for all $i \in \{1, \ldots, k\}$ such that $\mathbf{x}_i' \in span(\mathbf{x}_1, \ldots, \mathbf{x}_k)$.

Lemma 3

$$|G| \leq 2^k k!$$

Proof Let $g \in G$ be arbitrary. From Lemma 1, we have that $g(\mathbf{1}) = \mathbf{1}$, and $g(\mathbf{x}_i) = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_k \mathbf{x}_k$ for $i = 1, \ldots, k$. Now by Lemma 2 we know there exists a permutation $\sigma \in S_k$ such that $\mathbf{x}_i' = \pm \mathbf{x}_{\sigma(i)}$ for $i = 1, \ldots, k$. That is, g is essentially a signed permutation of the k main effects, so g is one of at most $2^k k!$ elements in G .

Next, by finding a subgroup of G that attains the upper bound on size, we determine the size and the structure of G . We first introduce some terminology and notation as given by Rotman [6] on pages 167 and 172.

Definition Let K be a (not necessarily normal) subgroup of a group G . Then a subgroup $Q \leq G$ is a **complement** of K in G if $K \cap Q = 1$ and $KQ = G$.

Definition A group G is a **semidirect product** of K by Q , denoted by $G = K \rtimes Q$, if $K \trianglelefteq G$ and K has a complement $Q' \cong Q$. One also says that G **splits** over K .

Definition Let D and Q be groups, let Ω be a finite Q-set, and let $K = \prod_{\omega \in \Omega} D_\omega$, where $D_\omega \cong D$ for all $\omega \in \Omega$. Then the **wreath product** of D by Q , denoted by $D \wr Q$ (or by D wr Q), is the semidirect product of K by Q , where Q acts on K by $q(d_\omega) = (d_{q\omega})$ for $q \in Q$ and $(d_\omega) \in \prod_{\omega \in \Omega} D_\omega$. The normal subgroup K of $D \wr Q$ is called the **base** of the wreath product .

In the factorial desing setting, S_k is the permutation group of k factors. The multiplicative group $\{\pm 1\}$ that multiplies columns is isomorphic to S_2 . Naturally, S_2^k is the direct product of k copies of S_2 . We now see that S_2^k is the base of $S_2 \wr S_k$.

Lemma 4

$$S_2^k \trianglelefteq S_2 \wr S_k$$

Proof Per the definition of the wreath product, S_k acts on S_2^k by permuting the entries of each $\{\pm 1\}^k$ vector in S_2^k . Let $\phi \in S_2^k$ and $\sigma \in S_k$ be arbitrary. Clearly, $\sigma\phi\sigma^{-1} \in S_2^k$. Furthermore, $\sigma^{-1}\phi\sigma \in S_2^k$, and $\sigma(\sigma^{-1}\phi\sigma)\sigma^{-1} = \phi$. Thus, $\sigma S_2^k \sigma^{-1} = S_2^k$ for all $\sigma \in S_k$, so $S_2^k \trianglelefteq S_2^k \rtimes S_k = S_2 \wr S_k$.

This wreath product is the set of all signed permutations of $\mathbf{x}_i$ for $i = 1, \ldots, k$ from the full factorial 2^k design, where $\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T$ constitute rows of $\mathbf{M}$. We shall see that this group is a subgroup of G . Hence, by Lemma 3, it is G .

Theorem 2

$$G = S_2 \wr S_k$$

Proof Consider an arbitrary element of $S_2 \wr S_k$. We know it can be written in the form $\phi\sigma$ where $\phi \in S_2^k$ and $\sigma \in S_k$ thanks to Lemma 4. Clearly, permuting the k rows $\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T$ of $\mathbf{M}$ will preserve the full factorial 2^k design as will negating any of these k rows. Hence, $\phi\sigma \in S_{2^k}$. Furthermore, the signed permutation $\phi\sigma$ clearly preserves $Row(\mathbf{M})$, so $\phi\sigma \in Aut(Row(\mathbf{M}))$. Because $\phi\sigma \in S_2 \wr S_k$ was arbitrary, we know $S_2 \wr S_k \leq G = Aut(Row(\mathbf{M})) \cap S_{2^k}$. Finally, $|S_2 \wr S_k| = |S_2^k||S_k| = 2^k k!$ which is the upper bound for $|G|$, so G must be exactly $S_2 \wr S_k$.

Corollary 1

$$|G| = 2^k k!$$

Proof As a direct result of Theorem 2, we have $|G| = |S_2 \wr S_k| = 2^k k!$.

3.3 The Symmetry Group for the Strength Two Case

From this point forward, we assume an OA$(N, k, 2, 2)$ in Equation 1. As in the strength one case, let

$$B = \{\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_k, \mathbf{x}_{1,2}, \ldots, \mathbf{x}_{k-1,k}\}$$

Once again, B is an orthogonal basis for $Row(\mathbf{M})$, and for all $g \in G$, we know $g(B)$ must also be an orthogonal basis for $Row(\mathbf{M})$ because $g \in Aut(Row(\mathbf{M})) \cap S_{2^k}$ as given by Theorem 1. Also, for every $\mathbf{x} \in B$, we know that $g(\mathbf{x})$ can be represented uniquely as a linear combination of the elements of B . In this case,

$$g(\mathbf{x}) = \lambda_0\mathbf{1} + \lambda_1\mathbf{x}_1 + \cdots + \lambda_k\mathbf{x}_k + \lambda_{1,2}\mathbf{x}_{1,2} + \cdots + \lambda_{k-1,k}\mathbf{x}_{k-1,k} \tag{3}$$

Similar to the strength one case above, we will arrive at the conclusion that for any $\mathbf{x} \in B$, every λ in Equation 3 must be zero except for one, which must have an absolute value of one. The following several lemmas serve to lead us to this conclusion.

Lemma 5 *Let* $\mathbf{x} \in B$. *If* $\mathbf{x} = \mathbf{1}$ *in Equation 3, then* $\lambda_0 = 1$, *and* $\lambda_i = 0$ *for* $i = 1, \ldots, k, (1,2), \ldots, (k-1,k)$. *Otherwise,* $\lambda_0 = 0$.

Proof Suppose $\mathbf{x} = \mathbf{1}$. Because $g \in G \leq S_{2^k}$, $g(\mathbf{1}) = \mathbf{1}$ which uniquely satisfies Equation 3. For $i = 1, \ldots, k, (1,2), \ldots, (k-1,k)$, we know $g(\mathbf{x}_i)$ must be orthogonal to $g(\mathbf{1}) = \mathbf{1}$, so $\lambda_0 = 0$ whenever $\mathbf{x} \neq \mathbf{1}$.

Lemma 6 *Let* $\mathbf{x} \in B$. *If* $\mathbf{x} \neq \mathbf{1}$ *in Equation 3, then* $\lambda_i \in \{0, \pm 0.5, \pm 1\}$ *for* $i = 1, \ldots, k, (1,2), \ldots, (k-1,k)$.

Proof Suppose $\mathbf{x} \neq \mathbf{1}$. Then Equation 3 becomes $g(\mathbf{x}) = \lambda_1 \mathbf{x}_1 + \cdots + \lambda_k \mathbf{x}_k + \lambda_{1,2}\mathbf{x}_{1,2} + \cdots + \lambda_{k-1,k}\mathbf{x}_{k-1,k}$. Because these basis vectors are the columns of the full factorial 2^k design and the corresponding 2-factor interactions obtained by taking the appropriate pairwise Hadamard products of the individual columns (main effects), we have the system of equations

$$\begin{array}{ccccccccccccc}
\lambda_1 & + & \cdots & +\lambda_k & +\lambda_{1,2} & + & \cdots & +\lambda_{1,k} & + & \cdots & +\lambda_{k-1,k} & = & \pm 1 \\
\vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & & \ddots & \vdots & & \vdots \\
\lambda_1 & - & \cdots & -\lambda_k & -\lambda_{1,2} & - & \cdots & -\lambda_{1,k} & + & \cdots & +\lambda_{k-1,k} & = & \pm 1 \\
-\lambda_1 & + & \cdots & +\lambda_k & -\lambda_{1,2} & - & \cdots & -\lambda_{1,k} & + & \cdots & +\lambda_{k-1,k} & = & \pm 1 \\
\vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots & & \ddots & \vdots & & \vdots \\
-\lambda_1 & - & \cdots & -\lambda_k & +\lambda_{1,2} & + & \cdots & +\lambda_{1,k} & + & \cdots & +\lambda_{k-1,k} & = & \pm 1
\end{array}$$

Subtracting the last equation from the first gives $\lambda_1 + \cdots + \lambda_k \in \{0, \pm 1\}$. Taking the difference of the middle equations likewise provides $\lambda_1 - \cdots - \lambda_k \in \{0, \pm 1\}$. Summing these two expressions results in the conclusion $\lambda_1 \in \{0, \pm 0.5, \pm 1\}$. Choosing other sets of equations similarly yields $\lambda_i \in \{0, \pm 0.5, \pm 1\}$ for $i = 1, \ldots, k, (1,2), \ldots, (k-1, k)$.

Lemma 7 *Let* $\mathbf{x} \in B$ *and* $g \in G$ *. If* $\mathbf{x} \neq \mathbf{1}$ *in Equation 3, then either* $g(\mathbf{x}) = \pm \mathbf{x}_i$ *for* $i \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$ *or* $g(\mathbf{x}) = \pm 0.5\mathbf{x}_a \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c \pm 0.5\mathbf{x}_d$ *for distinct* $a, b, c, d \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$.

Proof Suppose $\mathbf{x} \neq \mathbf{1}$. Because B is an orthogonal set, the Pythagorean Theorem gives

$$\begin{aligned} \|g(\mathbf{x})\|^2 &= \sum_i \|\lambda_i \mathbf{x}_i\|^2 \\ &= \sum_i (|\lambda_i| \|\mathbf{x}_i\|)^2 \\ &= \sum_i |\lambda_i|^2 \|\mathbf{x}_i\|^2 \\ &= \sum_i \lambda_i^2 \|\mathbf{x}_i\|^2 \end{aligned}$$

We note that $g \in S_{2^k}$ is norm-preserving, so $\|g(\mathbf{x})\| = 2^k = \|\mathbf{x}_i\|$ for $i = 1, \ldots, k, (1,2), \ldots, (k-1,k)$. Thus,

$$\sum_i \lambda_i^2 = 1$$

Clearly, not every λ_i can be zero. If $\lambda_i \in \{\pm 1\}$ for some $i \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$, then $\lambda_j = 0$ for all $j \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$ such that $i \neq j$. Otherwise, there must be distinct $a, b, c, d \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$ such that $\lambda_a, \lambda_b, \lambda_c, \lambda_d \in \{\pm 0.5\}$, and every other λ is zero. That is, either $g(\mathbf{x}) = \pm \mathbf{x}_i$ for $i \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$ or $g(\mathbf{x}) = \pm 0.5\mathbf{x}_a \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c \pm 0.5\mathbf{x}_d$ for distinct $a, b, c, d \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$.

Lemma 8 *Let* $\mathbf{x} \in B$ *and* $g \in G$ *. If* $g(\mathbf{x})$ *is of the second form given in Lemma 7,* *then* $g(\mathbf{x}) = \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ *for distinct* $a, b, c \in \{1, \ldots, k\}$.

Proof Suppose $g(\mathbf{x}) = \pm 0.5\mathbf{x}_a \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c \pm 0.5\mathbf{x}_d$ for distinct $a, b, c, d \in \{1, \ldots, k, (1,2), \ldots, (k-1, k)\}$. Clearly, $\mathbf{x}_a$, $\mathbf{x}_b$, $\mathbf{x}_c$, and $\mathbf{x}_d$ cannot all be main effects, for the full factorial design will ensure some entry of $g(\mathbf{x})$ equals $2 \notin \{\pm 1\}$. Therefore, at least one 2-factor interaction must be present in the linear combination. Because there are more such linear combinations than would be prudent to check manually, we take advantage of R software [5] at this stage of the proof. The code used for this step is contained in the appendix. By creating every essentially unique linear combination containing at least one 2-factor interaction term and checking whether they satisfy a basic requirement, we rule out all possibilities except those of one particular form. Specifically, by ruling out each linear combination where the minimum and maximum entries in the resulting vector are not -1 and 1 , respectively, we eliminate all linear combinations except those of the form $\pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$.

It is clear that $k \geq 3$ in order for the form in Lemma 8 to be viable.

Lemma 9 *Let* $\mathbf{x} \in B$ *and* $g \in G$. *If for every* $i \in \{1, \ldots, k\}$, $g(\mathbf{x}_i) \neq \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ *for distinct* $a, b, c \in \{1, \ldots, k\}$, *then* $g(\mathbf{x})$ *cannot be of the form in Lemma 8.*

Proof Recall from Lemma 5 that $g(\mathbf{1}) = \mathbf{1}$. Suppose that for every $i \in \{1, \ldots, k\}$, we have $g(\mathbf{x}_i) \neq \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$. Then we know from Lemmas 7 and 8 that for every $i \in \{1, \ldots, k\}$, there exists some $j \in \{1, \ldots, k, (1,2), \ldots, (k-1, k)\}$ such that $g(\mathbf{x}_i) = \pm \mathbf{x}_j$. Because g preserves Hadamard products, for every $i \in \{(1,2), \ldots, (k-1, k)\}$, there exists some $j \in \{1, \ldots, k, (1,2), \ldots, (k-1, k)\}$ such that $g(\mathbf{x}_i) = \pm \mathbf{x}_j$. Hence, for every $\mathbf{x} \in B$, $g(\mathbf{x}) \neq \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$.

Lemma 10 *Let $g \in G$. If for some $i \in \{1, \ldots, k\}$, $g(\mathbf{x}_i) = \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$, then there must exist some $j \in \{1, \ldots, k\}$ with $i \neq j$ such that $g(\mathbf{x}_j) = \pm 0.5\mathbf{x}_{a',b'} \pm 0.5\mathbf{x}_{a',c'} \pm 0.5\mathbf{x}_{b'} \pm 0.5\mathbf{x}_{c'}$ for distinct $a', b', c' \in \{1, \ldots, k\}$.*

Proof Suppose there exists some $i \in \{1, \ldots, k\}$, $g(\mathbf{x}_i) = \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$. By way of contradiction, suppose there is no $j \in \{1, \ldots, k\}$ with $i \neq j$ such that $g(\mathbf{x}_j) = \pm 0.5\mathbf{x}_{a',b'} \pm 0.5\mathbf{x}_{a',c'} \pm 0.5\mathbf{x}_{b'} \pm 0.5\mathbf{x}_{c'}$ for distinct $a', b', c' \in \{1, \ldots, k\}$. Then from Lemma 6 we know that for every $j \in \{1, \ldots, k\}$ with $i \neq j$, there exists some $l \in \{1, \ldots, k, (1, 2), \ldots, (k-1, k)\}$ such that $g(\mathbf{x}_j) = \pm \mathbf{x}_l$. Because g preserves Hadamard products, $g(\mathbf{x}_i) \circ \pm \mathbf{x}_l$ must also take on a viable form, and this implies that $\mathbf{x}_l \in \{\mathbf{x}_a, \mathbf{x}_{b,c}\}$. There can only be one such $\mathbf{x}_j$ because if there were more than one, their Hadamard product would be sent to something in $\{\pm \mathbf{1}, \pm \mathbf{x}_{a,b,c}\}$. But that means only two main effects ($\mathbf{x}_i$ and $\mathbf{x}_j$) get sent to viable forms, which contradicts Lemma 7. Thus, there must exist some $j \in \{1, \ldots, k\}$ with $i \neq j$ such that $g(\mathbf{x}_j) = \pm 0.5\mathbf{x}_{a',b'} \pm 0.5\mathbf{x}_{a',c'} \pm 0.5\mathbf{x}_{b'} \pm 0.5\mathbf{x}_{c'}$ for distinct $a', b', c' \in \{1, \ldots, k\}$.

Lemma 11 *Let $g \in G$. If there exist distinct $i, j \in \{1, \ldots, k\}$ such that $g(\mathbf{x}_i) = \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$ and $g(\mathbf{x}_j) = \pm 0.5\mathbf{x}_{a',b'} \pm 0.5\mathbf{x}_{a',c'} \pm 0.5\mathbf{x}_{b'} \pm 0.5\mathbf{x}_{c'}$ for distinct $a', b', c' \in \{1, \ldots, k\}$, then $\{a, b, c\} = \{a', b', c'\}$.*

Proof Suppose there exist distinct $i, j \in \{1, \ldots, k\}$ such that $g(\mathbf{x}_i) = \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$ and $g(\mathbf{x}_j) = \pm 0.5\mathbf{x}_{a',b'} \pm 0.5\mathbf{x}_{a',c'} \pm 0.5\mathbf{x}_{b'} \pm 0.5\mathbf{x}_{c'}$ for distinct $a', b', c' \in \{1, \ldots, k\}$. We proceed by way of contradiction and suppose that $\{a, b, c\} \neq \{a', b', c'\}$. That is, $|\{a, b, c\} \cap \{a', b', c'\}| < 3$. We observe that

$$
\begin{aligned}
g(\mathbf{x}_{i,j}) = \quad & \pm 0.25\mathbf{x}_{a,b,a',b'} && \pm 0.25\mathbf{x}_{a,b,a',c'} && \pm 0.25\mathbf{x}_{a,b,b'} && \pm 0.25\mathbf{x}_{a,b,c'} \\
& \pm 0.25\mathbf{x}_{a,c,a',b'} && \pm 0.25\mathbf{x}_{a,c,a',c'} && \pm 0.25\mathbf{x}_{a,c,b'} && \pm 0.25\mathbf{x}_{a,c,c'} \\
& \pm 0.25\mathbf{x}_{b,a',b'} && \pm 0.25\mathbf{x}_{b,a',c'} && \pm 0.25\mathbf{x}_{b,b'} && \pm 0.25\mathbf{x}_{b,c'} \\
& \pm 0.25\mathbf{x}_{c,a',b'} && \pm 0.25\mathbf{x}_{c,a',c'} && \pm 0.25\mathbf{x}_{c,b'} && \pm 0.25\mathbf{x}_{c,c'}
\end{aligned} \tag{4}
$$

(Case 1: $|\{a,b,c\} \cap \{a',b',c'\}| = 0$) Equation 4 clearly is not of a valid form.

(Case 2: $|\{a,b,c\} \cap \{a',b',c'\}| = 1$) If $a \neq a'$, 4-factor interaction terms will remain in Equation 4, so it will not be of a valid form. Suppose $a = a'$. Even if the 3-factor interaction terms were to cancel, the remaining 2-factor interaction terms are insufficient for Equation 4 to be of a valid form.

(Case 3: $|\{a,b,c\} \cap \{a',b',c'\}| = 2$) If $a \neq a'$, at least one 4-factor interaction term will remain in Equation 4, so it will not be of a valid form. Suppose $a = a'$. Without loss of generality, also suppose $b = b'$. Even if the 3-factor interaction terms and the $\mathbf{1}$ terms were to cancel, the remaining 2-factor interaction terms and main effect terms are insufficient for Equation 4 to be of a valid form.

This contradicts Equation 4 being of a valid form, so we conclude that $\{a,b,c\} = \{a',b',c'\}$.

Lemma 12 *If* $k \geq 4$ *, and if* $\mathbf{x} \neq \mathbf{1}$ *in Equation 3, then* $g(\mathbf{x}) = \pm\mathbf{x}_i$ *for* $i \in \{1,\ldots,k,(1,2),\ldots,(k-1,k)\}$.

Proof Let $k \geq 4$, and $\mathbf{x} \neq \mathbf{1}$ in Equation 3. By way of contradiction, suppose $g(\mathbf{x}) = \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a,b,c \in \{1,\ldots,k\}$. From Lemma 9, we know there must be some $i \in \{1,\ldots,k\}$ such that $g(\mathbf{x}_i) = \pm 0.5\mathbf{x}_{a',b'} \pm 0.5\mathbf{x}_{a',c'} \pm 0.5\mathbf{x}_{b'} \pm 0.5\mathbf{x}_{c'}$ for distinct $a',b',c' \in \{1,\ldots,k\}$. Lemma 10 guarantees there will be another main effect sent to a similar form by g , and Lemma 11 tells us it will be built from the same three distinct main effects and their three distinct 2-factor interactions.

At most one main effect could be sent by g to a form other than that just described as noted in the proof of Lemma 10. Now we have at least one more main effect to consider, and it must be sent to a form similar to that given above and also built from the same three main effects and their three distinct 2-factor interactions. But now we have four main effects that are sent to linear combinations of six orthogonal vectors, and the six resulting 2-factor interactions will necessarily also be sent by g to linear combinations of those same six orthogonal vectors (owing to the properties of the Hadamard product). This means that the ten new vectors cannot all be orthogonal, which contradicts $g \in G$. Hence, $g(\mathbf{x}) \neq \pm 0.5\mathbf{x}_{a,b} \pm 0.5\mathbf{x}_{a,c} \pm 0.5\mathbf{x}_b \pm 0.5\mathbf{x}_c$ for distinct $a, b, c \in \{1, \ldots, k\}$. Now from Lemmas 8 and 7, we have $g(\mathbf{x}) = \pm\mathbf{x}_i$ for $i \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$.

Lemma 13 *Let $k \geq 4$. Then*

$$|G| \leq 2^k (k+1)!$$

Proof Let $g \in G$ be arbitrary. Note that because g preserves Hadamard products, knowing how it acts on the main effects will determine how it acts on all of B . From Lemma 12, we know $g(\mathbf{x_1}) = \pm\mathbf{x}_i$ for $i \in \{1, \ldots, k, (1,2), \ldots, (k-1,k)\}$. Because $g(\mathbf{x}_{1,2})$ must be of a similar form, the possibilities for $g(\mathbf{x}_2)$ are restricted depending upon $g(\mathbf{x}_1)$. If $g(\mathbf{x}_1) = \pm\mathbf{x}_i$ for $i \in \{1, \ldots, k\}$, then $g(\mathbf{x}_2) = \pm\mathbf{x}_l$ for $l \in \{1, \ldots, i-1, i+1, \ldots, k, (1,i), \ldots, (i-1,i), (i,i+1), \ldots, (i,k)\}$. Otherwise, $g(\mathbf{x}_1) = \pm\mathbf{x}_{i,j}$ for $i < j$ and $i, j \in \{1, \ldots, k\}$, so $g(\mathbf{x}_2) = \pm\mathbf{x}_l$ for $l \in \{i, j, (1,i), (1,j), \ldots, (i-1,i), (i-1,j), (i,i+1), (i+1,j), \ldots, (i,j-1), (j-1,j), (i,j+1), (j,j+1), \ldots, (i,k), (j,k)\}$. To determine how many distinct possibilities exist, we shall consider four cases, based on the forms of $g(\mathbf{x}_1)$ and $g(\mathbf{x}_2)$, respectively.

(Case 1: main effect, main effect) Suppose $g(\mathbf{x}_1) = \pm\mathbf{x}_i$ for $i \in \{1, \ldots, k\}$ and

$g(\mathbf{x}_2) = \pm\mathbf{x}_l$ for $l \in \{1, \ldots, i-1, i+1, \ldots, k\}$. Then there are $2k$ possibilities for $g(\mathbf{x}_1)$ and $2(k-1)$ for $g(\mathbf{x}_2)$. All of the $(k-2)$ remaining main effects must be sent to plus or minus the other $(k-2)$ main effects. That is, there are $(2k)(2(k-1))(2^{k-2}(k-2)!) = 2^k k!$ distinct possibilities.

(Case 2: main effect, 2-factor interaction) Suppose $g(\mathbf{x}_1) = \pm\mathbf{x}_i$ for $i \in \{1, \ldots, k\}$ and $g(\mathbf{x}_2) = \pm\mathbf{x}_l$ for $l \in \{(1,i), \ldots, (i-1,i), (i,i+1), \ldots, (i,k)\}$. Then there are $2k$ possibilities for $g(\mathbf{x}_1)$ and $2(k-1)$ for $g(\mathbf{x}_2)$. All of the $(k-2)$ remaining main effects must be sent to plus or minus the other $(k-2)$ viable 2-factor interactions. That is, there are $(2k)(2(k-1))(2^{k-2}(k-2)!) = 2^k k!$ distinct possibilities.

(Case 3: 2-factor interaction, main effect) Suppose $g(\mathbf{x}_1) = \pm\mathbf{x}_{i,j}$ for $i < j$ and $i, j \in \{1, \ldots, k\}$ and $g(\mathbf{x}_2) = \pm\mathbf{x}_l$ for $l \in \{i, j\}$. Then there are $2\binom{k}{2}$ possibilities for $g(\mathbf{x}_1)$ and $2(2)$ for $g(\mathbf{x}_2)$. All of the $(k-2)$ remaining main effects must be sent to plus or minus the other $(k-2)$ viable 2-factor interactions. That is, there are $(2\binom{k}{2})(2(2))(2^{k-2}(k-2)!) = 2^k\binom{k}{2}(2)(k-2)!$ distinct possibilities.

(Case 4: 2-factor interaction, 2-factor interaction) Suppose $g(\mathbf{x}_1) = \pm\mathbf{x}_{i,j}$ for $i < j$ and $i, j \in \{1, \ldots, k\}$ and $g(\mathbf{x}_2) = \pm\mathbf{x}_l$ for $l \in \{(1,i), (1,j), \ldots, (i-1,i), (i-1,j), (i,i+1), (i+1,j), \ldots, (i,j-1), (j-1,j), (i,j+1), (j,j+1), \ldots, (i,k), (j,k)\}$. Then there are $2\binom{k}{2}$ possibilities for $g(\mathbf{x}_1)$ and $2(2k-4)$ for $g(\mathbf{x}_2)$. All of the $(k-2)$ remaining main effects must be sent to plus or minus the other $(k-3)$ viable 2-factor interactions and the lone viable main effect $\mathbf{x}_{\{i,j\}\cap l}$. That is, there are $(2\binom{k}{2})(2(2k-4))(2^{k-2}(k-2)!) = 2^k\binom{k}{2}(2k-4)(k-2)!$ distinct possibilities.

Therefore, the total number of possibilities for all cases is

$$\begin{aligned}
2^k k! &+ 2^k k! + 2^k \binom{k}{2}(2)(k-2)! + 2^k \binom{k}{2}(2k-4)(k-2)! \\
&= 2^k (k-2)! \left(2\left[k(k-1) + \binom{k}{2} + \binom{k}{2}(k-2)\right]\right) \\
&= 2^k (k-2)! \left(2\left[k(k-1) + \binom{k}{2}(k-1)\right]\right) \\
&= 2^k (k-1)! \left(2\left[k + \binom{k}{2}\right]\right) \\
&= 2^k (k-1)!(2k + k(k-1)) \\
&= 2^k (k)!(2 + k - 1) \\
&= 2^k (k+1)!
\end{aligned}$$

Thus, g is one of at most $2^k(k+1)!$ elements in G .

Theorem 3 *Let $k \geq 4$. Then*

$$G = S_2^k \rtimes S_{k+1}$$

Proof Let $R = \langle \rho_1, \ldots, \rho_k \rangle$ where ρ_i acts on the full factorial design by sending $(\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_k)$ to $(\mathbf{x}_{1,i}, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_{i,k})$ for $i = 1, \ldots, k$. Note that elements of R preserve the full factorial design as well as $Row(\mathbf{M})$, so $R \leq G$. Furthermore, for $i = 1, \ldots, k$, $\rho_i^{-1} = \rho_i$. For any distinct $i, j \in \{1, \ldots, k\}$, $\rho_i \rho_j \rho_i$ simply permutes $\mathbf{x}_i$ and $\mathbf{x}_j$ within the full factorial design, so clearly $S_k \leq R$. Now we see that $\rho_j \rho_i S_k = \rho_i S_k$ for distinct $i, j \in \{1, \ldots, k\}$, so there are exactly $k+1$ left cosets of S_k within R , and together these constitute the entirety of R . Hence, $R \cong S_{k+1}$. Letting $\phi \in S_2^k$ be arbitrary, we note that for any $i = 1, \ldots, k$, $\rho_i^{-1} = \rho_i$, $\rho_i^{-1}\phi\rho_i = \phi'$ where $\phi' \in S_2^k$, and $\rho_i^{-1}\phi'\rho_i = \phi$. Together with this information, Lemma 4 makes it clear that $S_2^k \trianglelefteq S_2^k \rtimes S_{k+1}$. Now $S_2^k \rtimes S_{k+1} \leq G$, and $|S_2^k \rtimes S_{k+1}| = |S_2^k||S_{k+1}| = 2^k(k+1)!$

which is the upper bound for $|G|$, so G must be exactly $S_2^k \rtimes S_{k+1}$.

Corollary 2 *Let $k \geq 4$. Then*

$$|G| = 2^k(k+1)!$$

Proof As a direct result of Theorem 3, we have $|G| = |S_2^k \rtimes S_{k+1}| = 2^k(k+1)!$.

Note that when $k = 3$, the result above does not hold. For example, consider the permutation $g \in G$ such that

$$\begin{aligned} g(\mathbf{x}_1) &= 0.5\mathbf{x}_{1,2} + 0.5\mathbf{x}_{1,3} + 0.5\mathbf{x}_2 - 0.5\mathbf{x}_3 \\ g(\mathbf{x}_2) &= 0.5\mathbf{x}_{1,2} + 0.5\mathbf{x}_{1,3} - 0.5\mathbf{x}_2 + 0.5\mathbf{x}_3 \\ g(\mathbf{x}_3) &= \mathbf{x}_1 \end{aligned}$$

Because this permutation sends main effects to forms other than those which were viable for $k \geq 4$, we conclude $|G| > 2^3(3{+}1)! = 192$. This observation is corroborated by the Geyer [2] algorithm and GAP [1], which prove that in this case $|G| = 1152$, and $G \cong (S_4 \times S_4) \rtimes S_2$.

IV. TOPICS FOR FUTURE RESEARCH

It is easy to see that $S_2 \wr S_k$ is always a subgroup of G , and $S_2^k \rtimes S_{k+1}$ is a subgroup when t is even. Perhaps the entirety of G can be found for $\mathrm{OA}(N, k, 2, t)$ with $t > 2$. Also, finding the LP relaxation symmetry group of the ILP formulation in [3] of an $\mathrm{OA}(N, k, s, t)$ for $s > 2$ is an open problem. In this case, it is easy to see that $S_s \wr S_k$ is always a subgroup of this group.

Appendix A. R CODE AND CASES

```
################################################################################
# Function-  expandcases - generates linear combinations with positive 1st term
# Input(s)-  lst - binary representation of main effects and interactions
# Output(s)- newlst - linear combinations with positive 1st term
################################################################################

expandcases<-function(lst){
ll<-length(lst)
newlst<-list()
r<-1
aa<-as.matrix(expand.grid(c(1,-1),c(1,-1),c(1,-1)))
aa<-cbind(1,aa)
for (j in 1:ll){
for (i in 1:8){
aa2<-rbind(aa[i,],lst[[j]])
 dimnames(aa2)[[2]]<-NULL
newlst[r]<-list(aa2)
r<-r+1
}}
return(newlst)
}

################################################################################
# Function-  allcheckbinvector - checks viability of linear combinations
# Input(s)-  lst - all linear combinations to be checked
# Output(s)- displays 1s for viable combinations and 0s otherwise
################################################################################

allcheckbinvector<-function(lst){
l<-length(lst)
for (i in 1:l){
print(checkbinaryvector(lst[[i]]))
}
}

################################################################################
# Function-  checkbinaryvector - checks viability of a linear combination
# Input(s)-  newcik - the linear combination to be checked
# Output(s)- 1 if combination is viable, 0 otherwise
################################################################################

checkbinaryvector<-function(newcik){
newcikmat<-newcik[-1,]
newcikcoff<-newcik[1,]
pp<-dim(newcikmat)
cols<-pp[2]
pp<-pp[1]
full<-c("expand.grid(c(1,-1)")
for (j in 1:(pp-1)){
full<-paste(full,",c(1,-1)")
}
full<-paste(full,")")
full<-parse(text = full)
full<-eval(full)

temp<-genprod(full,newcikmat[,1])

for (i in 2:cols){
temp<-cbind(temp,genprod(full,newcikmat[,i]))
}

finvec<-as.matrix(temp[,1]*newcikcoff[1])

for (i in 2:cols){
finvec<-finvec+newcikcoff[i]*temp[,i]
}
finvec<-as.matrix(as.integer(finvec/2))

if(min(finvec)==-1 & max(finvec)==1){
return(1)} else{return(0)}
}

################################################################################
# Function-  genprod - computes Hadamard product
# Input(s)-  full - 2-level (+/-1) full factorial design
#            tt1 - indicator of which main effects are to be multiplied
# Output(s)- as.matrix(outfullcheck) - +/-1 form of main effect or interaction
################################################################################

genprod<-function(full,tt1){
outfullcheck<-full[,1]^(tt1[1])
pp<-length(tt1)
for (j in 1:(pp-1)){
outfullcheck<-outfullcheck*full[,(j+1)]^(tt1[(j+1)])
}
return(as.matrix(outfullcheck))
}

################################################################################
# Cases with 1 two-factor interaction
################################################################################

a12.1.2.3<-cbind(c(1,1,0),c(1,0,0),c(0,1,0),c(0,0,1))
c12.1.2.3<-expandcases(list(a12.1.2.3))
allcheckbinvector(c12.1.2.3)

a12.1.3.4<-cbind(c(1,1,0,0),c(1,0,0,0),c(0,0,1,0),c(0,0,0,1))
c12.1.3.4<-expandcases(list(a12.1.3.4))
allcheckbinvector(c12.1.3.4)
```

```
a12.3.4.5<-cbind(c(1,1,0,0,0),c(0,0,1,0,0),c(0,0,0,1,0),c(0,0,0,0,1))
c12.3.4.5<-expandcases(list(a12.3.4.5))
allcheckbinvector(c12.3.4.5)

################################################################################
# Cases with 2 two-factor interactions
################################################################################

a12.13<-cbind(c(1,1,0,0,0,0),c(1,0,1,0,0,0))
a12.34<-cbind(c(1,1,0,0,0,0),c(0,0,1,1,0,0))

a12.13.1<-cbind(a12.13,c(1,0,0,0,0,0))
a12.13.2<-cbind(a12.13,c(0,1,0,0,0,0))
a12.13.4<-cbind(a12.13,c(0,0,0,1,0,0))
a12.34.1<-cbind(a12.34,c(1,0,0,0,0,0))
a12.34.5<-cbind(a12.34,c(0,0,0,0,1,0))

a12.13.1.2<-cbind(a12.13.1,c(0,1,0,0,0,0))
a12.13.1.2<-a12.13.1.2[c(1:3),]
c12.13.1.2<-expandcases(list(a12.13.1.2))
allcheckbinvector(c12.13.1.2)

a12.13.1.4<-cbind(a12.13.1,c(0,0,0,1,0,0))
a12.13.1.4<-a12.13.1.4[c(1:4),]
c12.13.1.4<-expandcases(list(a12.13.1.4))
allcheckbinvector(c12.13.1.4)

a12.13.2.3<-cbind(a12.13.2,c(0,0,1,0,0,0))
a12.13.2.3<-a12.13.2.3[c(1:3),]
c12.13.2.3<-expandcases(list(a12.13.2.3))
allcheckbinvector(c12.13.2.3)

a12.13.2.4<-cbind(a12.13.2,c(0,0,0,1,0,0))
a12.13.2.4<-a12.13.2.4[c(1:4),]
c12.13.2.4<-expandcases(list(a12.13.2.4))
allcheckbinvector(c12.13.2.4)

a12.13.4.5<-cbind(a12.13.4,c(0,0,0,0,1,0))
a12.13.4.5<-a12.13.4.5[c(1:5),]
c12.13.4.5<-expandcases(list(a12.13.4.5))
allcheckbinvector(c12.13.4.5)

a12.34.1.2<-cbind(a12.34.1,c(0,1,0,0,0,0))
a12.34.1.2<-a12.34.1.2[c(1:4),]
c12.34.1.2<-expandcases(list(a12.34.1.2))
allcheckbinvector(c12.34.1.2)

a12.34.1.3<-cbind(a12.34.1,c(0,0,1,0,0,0))
a12.34.1.3<-a12.34.1.3[c(1:4),]
c12.34.1.3<-expandcases(list(a12.34.1.3))
allcheckbinvector(c12.34.1.3)
```

```
a12.34.1.5<-cbind(a12.34.1,c(0,0,0,0,1,0))
a12.34.1.5<-a12.34.1.5[c(1:5),]
c12.34.1.5<-expandcases(list(a12.34.1.5))
allcheckbinvector(c12.34.1.5)

a12.34.5.6<-cbind(a12.34.5,c(0,0,0,0,0,1))
a12.34.5.6<-a12.34.5.6[c(1:6),]
c12.34.5.6<-expandcases(list(a12.34.5.6))
allcheckbinvector(c12.34.5.6)

################################################################################
# Cases with 3 two-factor interactions
################################################################################

a12.13<-cbind(c(1,1,0,0,0,0,0),c(1,0,1,0,0,0,0))
a12.34<-cbind(c(1,1,0,0,0,0,0),c(0,0,1,1,0,0,0))

a12.13.14<-cbind(a12.13,c(1,0,0,1,0,0,0))
a12.13.23<-cbind(a12.13,c(0,1,1,0,0,0,0))
a12.13.24<-cbind(a12.13,c(0,1,0,1,0,0,0))
a12.13.45<-cbind(a12.13,c(0,0,0,1,1,0,0))
a12.34.56<-cbind(a12.34,c(0,0,0,0,1,1,0))

a12.13.14.1<-cbind(a12.13.14,c(1,0,0,0,0,0,0))
a12.13.14.1<-a12.13.14.1[c(1:4),]
c12.13.14.1<-expandcases(list(a12.13.14.1))
allcheckbinvector(c12.13.14.1)

a12.13.14.2<-cbind(a12.13.14,c(0,1,0,0,0,0,0))
a12.13.14.2<-a12.13.14.2[c(1:4),]
c12.13.14.2<-expandcases(list(a12.13.14.2))
allcheckbinvector(c12.13.14.2)

a12.13.14.5<-cbind(a12.13.14,c(0,0,0,0,1,0,0))
a12.13.14.5<-a12.13.14.5[c(1:5),]
c12.13.14.5<-expandcases(list(a12.13.14.5))
allcheckbinvector(c12.13.14.5)

a12.13.23.1<-cbind(a12.13.23,c(1,0,0,0,0,0,0))
a12.13.23.1<-a12.13.23.1[c(1:3),]
c12.13.23.1<-expandcases(list(a12.13.23.1))
allcheckbinvector(c12.13.23.1)

a12.13.23.4<-cbind(a12.13.23,c(0,0,0,1,0,0,0))
a12.13.23.4<-a12.13.23.4[c(1:4),]
c12.13.23.4<-expandcases(list(a12.13.23.4))
allcheckbinvector(c12.13.23.4)

a12.13.24.1<-cbind(a12.13.24,c(1,0,0,0,0,0,0))
a12.13.24.1<-a12.13.24.1[c(1:4),]
c12.13.24.1<-expandcases(list(a12.13.24.1))
allcheckbinvector(c12.13.24.1)
```

```
a12.13.24.3<-cbind(a12.13.24,c(0,0,1,0,0,0,0))
a12.13.24.3<-a12.13.24.3[c(1:4),]
c12.13.24.3<-expandcases(list(a12.13.24.3))
allcheckbinvector(c12.13.24.3)

a12.13.24.5<-cbind(a12.13.24,c(0,0,0,0,1,0,0))
a12.13.24.5<-a12.13.24.5[c(1:5),]
c12.13.24.5<-expandcases(list(a12.13.24.5))
allcheckbinvector(c12.13.24.5)

a12.13.45.1<-cbind(a12.13.45,c(1,0,0,0,0,0,0))
a12.13.45.1<-a12.13.45.1[c(1:5),]
c12.13.45.1<-expandcases(list(a12.13.45.1))
allcheckbinvector(c12.13.45.1)

a12.13.45.2<-cbind(a12.13.45,c(0,1,0,0,0,0,0))
a12.13.45.2<-a12.13.45.2[c(1:5),]
c12.13.45.2<-expandcases(list(a12.13.45.2))
allcheckbinvector(c12.13.45.2)

a12.13.45.5<-cbind(a12.13.45,c(0,0,0,0,1,0,0))
a12.13.45.5<-a12.13.45.5[c(1:5),]
c12.13.45.5<-expandcases(list(a12.13.45.5))
allcheckbinvector(c12.13.45.5)

a12.13.45.6<-cbind(a12.13.45,c(0,0,0,0,0,1,0))
a12.13.45.6<-a12.13.45.6[c(1:6),]
c12.13.45.6<-expandcases(list(a12.13.45.6))
allcheckbinvector(c12.13.45.6)

a12.34.56.1<-cbind(a12.34.56,c(1,0,0,0,0,0,0))
a12.34.56.1<-a12.34.56.1[c(1:6),]
c12.34.56.1<-expandcases(list(a12.34.56.1))
allcheckbinvector(c12.34.56.1)

a12.34.56.7<-cbind(a12.34.56,c(0,0,0,0,0,0,1))
c12.34.56.7<-expandcases(list(a12.34.56.7))
allcheckbinvector(c12.34.56.7)

################################################################################
# Cases with 4 two-factor interactions
################################################################################

a12.13<-cbind(c(1,1,0,0,0,0,0,0),c(1,0,1,0,0,0,0,0))
a12.34<-cbind(c(1,1,0,0,0,0,0,0),c(0,0,1,1,0,0,0,0))

a12.13.14<-cbind(a12.13,c(1,0,0,1,0,0,0,0))
a12.13.23<-cbind(a12.13,c(0,1,1,0,0,0,0,0))
a12.13.24<-cbind(a12.13,c(0,1,0,1,0,0,0,0))
a12.13.45<-cbind(a12.13,c(0,0,0,1,1,0,0,0))
a12.34.56<-cbind(a12.34,c(0,0,0,0,1,1,0,0))
```

```
a12.13.14.15<-cbind(a12.13.14,c(1,0,0,0,1,0,0,0))
a12.13.14.15<-a12.13.14.15[c(1:5),]
c12.13.14.15<-expandcases(list(a12.13.14.15))
allcheckbinvector(c12.13.14.15)

a12.13.14.23<-cbind(a12.13.14,c(0,1,1,0,0,0,0,0))
a12.13.14.23<-a12.13.14.23[c(1:4),]
c12.13.14.23<-expandcases(list(a12.13.14.23))
allcheckbinvector(c12.13.14.23)

a12.13.14.25<-cbind(a12.13.14,c(0,1,0,0,1,0,0,0))
a12.13.14.25<-a12.13.14.25[c(1:5),]
c12.13.14.25<-expandcases(list(a12.13.14.25))
allcheckbinvector(c12.13.14.25)

a12.13.14.56<-cbind(a12.13.14,c(0,0,0,0,1,1,0,0))
a12.13.14.56<-a12.13.14.56[c(1:6),]
c12.13.14.56<-expandcases(list(a12.13.14.56))
allcheckbinvector(c12.13.14.56)

a12.13.23.45<-cbind(a12.13.23,c(0,0,0,1,1,0,0,0))
a12.13.23.45<-a12.13.23.45[c(1:5),]
c12.13.23.45<-expandcases(list(a12.13.23.45))
allcheckbinvector(c12.13.23.45)

a12.13.24.35<-cbind(a12.13.24,c(0,0,1,0,1,0,0,0))
a12.13.24.35<-a12.13.24.35[c(1:5),]
c12.13.24.35<-expandcases(list(a12.13.24.35))
allcheckbinvector(c12.13.24.35)

a12.13.24.56<-cbind(a12.13.24,c(0,0,0,0,1,1,0,0))
a12.13.24.56<-a12.13.24.56[c(1:6),]
c12.13.24.56<-expandcases(list(a12.13.24.56))
allcheckbinvector(c12.13.24.56)

a12.13.45.56<-cbind(a12.13.45,c(0,0,0,0,1,1,0,0))
a12.13.45.56<-a12.13.45.56[c(1:6),]
c12.13.45.56<-expandcases(list(a12.13.45.56))
allcheckbinvector(c12.13.45.56)

a12.13.45.67<-cbind(a12.13.45,c(0,0,0,0,0,1,1,0))
a12.13.45.67<-a12.13.45.67[c(1:7),]
c12.13.45.67<-expandcases(list(a12.13.45.67))
allcheckbinvector(c12.13.45.67)

a12.34.56.78<-cbind(a12.34.56,c(0,0,0,0,0,0,1,1))
c12.34.56.78<-expandcases(list(a12.34.56.78))
allcheckbinvector(c12.34.56.78)
```

Bibliography

1. GAP. 2013. *GAP – Groups, Algorithms, and Programming, Version 4.6.4*. The GAP Group.

2. Geyer, A.J. 2014. *Different Formulations of the Orthogonal Array Problem and Their Symmetries*. PhD dissertation, Air Force Institute of Technology.

3. Geyer, A.J., Bulutoglu, D.A., & Rosenberg, S.J. 2014. The LP Relaxation Orthogonal Array Polytope and its Permutation Symmetries. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **91**, 165–176.

4. Margot, F. 2010. Symmetry in Integer Linear Programming. *Pages 647–686 of: 50 Years of Integer Programming 1958-2008*. Springer.

5. R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

6. Rotman, J.J. 1994. *An Introduction to the Theory of Groups*. 4 edn. Springer.

7. Stufken, J., & Tang, B. 2007. Complete Enumeration of Two-Level Orthogonal Arrays of Strength d With d+2 Constraints. *The Annals of Statistics*, **35**(2), 793–814.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 26–03–2015	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* SEPT 2013 — MAR 2015

4. TITLE AND SUBTITLE Symmetry Groups for Linear Programming Relaxations of Orthogonal Array Problems	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Arquette, David M., Second Lieutenant, USAF	**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENC) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENC-MS-15-M-003
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) THIS SECTION INTENTIONALLY LEFT BLANK	**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A.
Approved for Public Release; distribution unlimited.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Integer linear programs arise in many situations, and solving such problems can be computationally demanding. One way to solve them more efficiently is by exploiting the symmetry within their formulation. This paper proves that the symmetry group for the linear programming relaxation of 2-level orthogonal array problems of strength 2 is a particular semidirect product.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. D. A. Bulutoglu, AFIT/ENC
U	U	U	UU	38	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-6565 x4704; dursun.bulutoglu@afit.edu