Fractional Factorial Designs by Combining Two-Level Designs

Alan Vázquez, Peter Goos & Eric Schoen

University of Antwerp

alan.vazquezalcocer@uantwerpen.be

ENBIS-15, 2015

Outline

1. Neural Network Training Experiment
2. Evaluating Experimental Designs
3. Construction Method
4. Results and Conclusions

Outline

Neural Network Training Experiment

- Investigate the architecture and training of a neural network to predict short-term load requirements for an Ohio electric utility (Mee, 2011)

Neural Network Training Experiment

- Investigate the architecture and training of a neural network to predict short-term load requirements for an Ohio electric utility (Mee, 2011)
- Ten factors under study each at two levels

Factors		Levels	
		-1	1
A	Hidden layers	1	2
B	Transfer function in output layer	Linear	Sigmoid
C	Transfer function in hidden layer	Sigmoid	Sinusoid
D	Backpropagation learning algorithm	Standard	Cumulative
E	Gaussian noise added	No	Yes
F	Stopping rule	RMSE	CD
G	Network	Feedforward	Recurrent
H	Years of training data	2	4
J	Time of peak	Winter	Summer
K	Industrial load	Low	High

Neural Network Training Experiment

- Investigate the architecture and training of a neural network to predict short-term load requirements for an Ohio electric utility (Mee, 2011)
- Ten factors under study each at two levels
- Budget allows for 64 observations

		Levels	
Factors		-1	1
A	Hidden layers	1	2
B	Transfer function in output layer	Linear	Sigmoid
C	Transfer function in hidden layer	Sigmoid	Sinusoid
D	Backpropagation learning algorithm	Standard	Cumulative
E	Gaussian noise added	No	Yes
F	Stopping rule	RMSE	CD
G	Network	Feedforward	Recurrent
H	Years of training data	2	4
J	Time of peak	Winter	Summer
K	Industrial load	Low	High

Neural Network Training Experiment

Prior information:

1. Most of the 10 factors will be active
2. Considerable number of active two-factor interactions
3. Three-factor and higher order interactions are negligible

Neural Network Training Experiment

Prior information:

1. Most of the 10 factors will be active
2. Considerable number of active two-factor interactions
3. Three-factor and higher order interactions are negligible

Design Problem:

Construct an efficient experimental plan

Experimental plans

Regular designs:

MA64 2^{10-4}_{IV} design

- $\mathbf{G} = \mathbf{BCDF}$, $\mathbf{H} = \mathbf{ABDE}$, $\mathbf{J} = \mathbf{ACDF}$, and $\mathbf{K} = \mathbf{ABCE}$
- Minimum Aberration design

Experimental plans

Regular designs:

MA64 2^{10-4}_{IV} design

- **G = BCDF**, **H = ABDE**, **J = ACDF**, and **K = ABCE**
- Minimum Aberration design

International Statistical Review (1993), **61**, 1, pp. 131–145. Printed in Great Britain

A Catalogue of Two-level and Three-level Fractional Factorial Designs with Small Runs

Jiahua Chen, D.X. Sun and C.F.J. Wu

Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

Experimental plans

Regular designs:

MA64 2_{IV}^{10-4} design

- $\mathbf{G} = \mathbf{BCDF}$, $\mathbf{H} = \mathbf{ABDE}$,
  $\mathbf{J} = \mathbf{ACDF}$, and $\mathbf{K} = \mathbf{ABCE}$
- Minimum Aberration design

International Statistical Review (1993), **61**, 1, pp. 131–145. Printed in Great Britain

A Catalogue of Two-level and Three-level Fractional Factorial Designs with Small Runs

Jiahua Chen, D.X. Sun and C.F.J. Wu

Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

R package **FrF2** (Grömping, 2014)

Experimental plans

Regular designs:

MA64 2^{10-4}_{IV} design

- $\mathbf{G} = \mathbf{BCDF}$, $\mathbf{H} = \mathbf{ABDE}$, $\mathbf{J} = \mathbf{ACDF}$, and $\mathbf{K} = \mathbf{ABCE}$
- Minimum Aberration design

Nonregular designs:

XW64 64-run design with 10 factors (Xu & Wong, 2007)

- Constructed from *Quaternary Linear Codes*

Orthogonal Arrays of Strength Three

The designs considered for our problem, regular and nonregular, belong to a class called *orthogonal arrays of strength three*.

Strength-3 OAs have the following properties:

1. Main effects are not correlated with two-factor interactions
2. Pairs of two-factor interactions can be correlated

Orthogonal Arrays of Strength Three

The designs considered for our problem, regular and nonregular, belong to a class called *orthogonal arrays of strength three*.

Strength-3 OAs have the following properties:

1. Main effects are not correlated with two-factor interactions
2. Pairs of two-factor interactions can be correlated

Fully correlated Regular designs

Partially correlated Nonregular designs

Outline

Evaluating Experimental Designs

How can we measure the correlation between pairs of two-factor interactions in strength-3 OAs?

Evaluating Experimental Designs

How can we measure the correlation between pairs of two-factor interactions in strength-3 OAs?

According to Deng & Tang (1999), the following holds:

- For the MA64 and XW64 designs, the possible absolute values for the correlation between pairs of 2fi's are 1, 0.75, 0.5, 0.25, 0.

Evaluating Experimental Designs

The F_4 vector has entries equal to the frequencies for the possible absolute correlation values between pairs of 2fi's, divided by three (Deng & Tang, 1999).

Evaluating Experimental Designs

The F_4 vector has entries equal to the frequencies for the possible absolute correlation values between pairs of 2fi's, divided by three (Deng & Tang, 1999).

Example:

MA64 design	XW64 design
$F_4(1, 0.75, 0.5, 0.25) = (2, 0, 0, 0)$	$F_4(1, 0.75, 0.5, 0.25) = (0, 0, 8, 0)$
6 pairs of 2fi's	24 pairs of 2fi's

Evaluating Experimental Designs

The F_4 vector has entries equal to the frequencies for the possible absolute correlation values between pairs of 2fi's, divided by three (Deng & Tang, 1999).

Example:

MA64 design	XW64 design
$F_4(1, 0.75, 0.5, 0.25) = (2, 0, 0, 0)$	$F_4(1, 0.75, 0.5, 0.25) = (0, 0, 8, 0)$
6 pairs of 2fi's	24 pairs of 2fi's

F_4

Sequentially minimizing F_4 is equivalent to minimizing: (1) the maximum absolute correlation between pairs of two-factor interactions; and, (2) the total number of pairs involved.

Evaluating Experimental Designs

Tang & Deng (1999) defined:

A_4

The A_4 value is proportional to the sum of squared correlations between pairs of two-factor interactions.

Evaluating Experimental Designs

Tang & Deng (1999) defined:

A_4

The A_4 value is proportional to the sum of squared correlations between pairs of two-factor interactions.

Example: For the MA64 design
$A_4 = 2(1)^2 + 0(0.75)^2 + 0(0.5)^2 + 0(0.25)^2 = 2$

while for the XW64 design
$A_4 = 0(1)^2 + 0(0.75)^2 + 8(0.5)^2 + 0(0.25)^2 = 2$

Evaluating Experimental Designs

The ability of a design to *estimate* two-factor interactions is measured as the rank(X_2) (Schoen & Mee, 2012).

- X_2 is the matrix containing the two-factor interactions

Evaluating Experimental Designs

The ability of a design to *estimate* two-factor interactions is measured as the rank(X_2) (Schoen & Mee, 2012).

- X_2 is the matrix containing the two-factor interactions

 # Estimable 2fi's

The higher the rank(X_2) of a design, the more two-factor interactions that can be estimated based on that design.

Evaluating Experimental Designs

Example:

MA64 design
rank$(X_2) = 39$

XW64 design
rank$(X_2) = 39$

The total number of interactions is $\binom{10}{2} = 45$. Therefore, there will be six 2fi's that cannot be included in the model.

Evaluating Experimental Designs

Discussion:

- Both designs provide the same number of estimable two-factor interactions and A_4 values
- XW64 design is preferred because provide less maximum absolute correlation
- However, the maximum absolute correlation between pairs of two-factor interactions is 0.5
- We cannot include all two-factor interactions in the model

Evaluating Experimental Designs

Discussion:

- Both designs provide the same number of estimable two-factor interactions and A_4 values
- XW64 design is preferred because provide less maximum absolute correlation
- However, the maximum absolute correlation between pairs of two-factor interactions is 0.5
- We cannot include all two-factor interactions in the model

Can we construct a better alternative design?

Outline

Construction by Example

1. Consider the 2^{8-4}_{IV} design as D_u and D_l

   16 runs and 8 factors

$$D = \left[\begin{array}{c} \left.\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right\} D_u \\ \hline \left.\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right\} D_l \end{array}\right]$$

32 runs and 8 factors

Construction by Example

1. Consider the 2^{8-4}_{IV} design as D_u and D_l

   16 runs and 8 factors

2. Consider foldover plans with column permutations of D_l to sequentially minimize the F_4 vector or minimize the A_4 value of D

$$D = \left[\begin{array}{cccccccc}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right]
\begin{array}{l}
\left.\rule{0pt}{16em}\right\} D_u \\
\left.\rule{0pt}{16em}\right\} D_l
\end{array}$$

32 runs and 8 factors

Construction by Example

1. Consider the 2^{8-4}_{IV} design as D_u and D_l

   16 runs and 8 factors

2. Consider foldover plans with column permutations of D_l to sequentially minimize the F_4 vector or minimize the A_4 value of D

3. Evaluating all possible combined designs D would require $8! \times 2^8 = 10,321,920$ evaluations

$$D = \left[\begin{array}{c}
\left.\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right\} D_u \\ \hline
\left.\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right\} D_l
\end{array}\right]$$

32 runs and 8 factors

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

$$D = \left[\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right]$$

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (14,\ 0)$

$$D = \left[\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}\right]$$

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (7,\ 0)$

Foldover column 7

$$D = \left[\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & -1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & 1
\end{array}\right]$$

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (6,\ 0)$

Foldover column 8

$$D = \left[\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 \\
-1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \\
-1 & -1 & 1 & -1 & 1 & -1 & -1 & -1 \\
-1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 \\
-1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 \\
1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & 1 & -1 & 1 & 1 & 1 \\
1 & 1 & 1 & -1 & 1 & -1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1
\end{array}\right]$$

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (2,\ 16)$

Swap columns 5 and 6

$$D = \left[\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 \\
-1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & 1 & -1 & -1 \\
-1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 \\
1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & -1
\end{array}\right]$$

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (0,\ 24)$

Swap columns 6 and 7

$$D = \left[\begin{array}{rrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
-1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 \\
-1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 \\
1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\
1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & -1 & 1 & -1
\end{array}\right]$$

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (0,\ 24)$

$$
D = \left[\begin{array}{ccccccccc}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
-1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
-1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
-1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
-1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \\
1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 \\
1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & -1
\end{array}\right]
$$

32 runs and 9 factors

An algorithmic approach

- Variable Neighborhood Search (VNS)
  – Framework to construct efficient algorithms
- Two *moves*: (1) foldover (2) swap columns

Example:

$F_4(1, 0.5) = (0,\ 24)$

Li & Lin (2015)
– regular designs
– the same design

$$D = \left[\begin{array}{rrrrrrrrr}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 \\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 \\
-1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
-1 & 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
-1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 \\
-1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \\
-1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 \\
1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 \\
1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
1 & -1 & 1 & 1 & 1 & -1 & 1 & -1 & -1 \\
1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \\
1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & -1 \\
1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & -1 & 1 & -1
\end{array}\right]$$

Outline

1. Neural Network Training Experiment
2. Evaluating Experimental Designs
3. Construction Method
4. Results and Conclusions

Results

- According to Schoen et al. (2010), there are 34 strength-3 OAs with 32 runs and 9 factors.
- We tested all possible combinations of OAs as D_u and D_l and used the proposed methodology.

Results

- According to Schoen et al. (2010), there are 34 strength-3 OAs with 32 runs and 9 factors.
- We tested all possible combinations of OAs as D_u and D_l and used the proposed methodology.

Design	$\{D_u, D_l\}$	$F_4(1, 0.75, 0.5, 0.25)$	A_4	Est. 2fi's
64.10	$\{27, 34\}$	(0, 0, 0, 32)	2	45

Results

- According to Schoen et al. (2010), there are 34 strength-3 OAs with 32 runs and 9 factors.
- We tested all possible combinations of OAs as D_u and D_l and used the proposed methodology.

Design	$\{D_u, D_l\}$	$F_4(1, 0.75, 0.5, 0.25)$	A_4	Est. 2fi's
64.10	$\{27, 34\}$	(0, 0, 0, 32)	2	45

Existing alternatives:

Design	$F_4(1, 0.75, 0.5, 0.25)$	A_4	Est. 2fi's
MA64	(2, 0, 0, 0)	2	39
XW64	(0, 0, 8, 0)	2	39

One last comparison

What if we want to use a D-optimal design?

One last comparison

What if we want to use a D-optimal design?

- A D-optimal design for a model containing the intercept, ME's and 2fi's (Atkinson et al., 2011)
- A D-optimal design with 64 runs and 10 factors was constructed using the R package **AlgDesign** (Wheeler, 2011)

One last comparison

What if we want to use a D-optimal design?

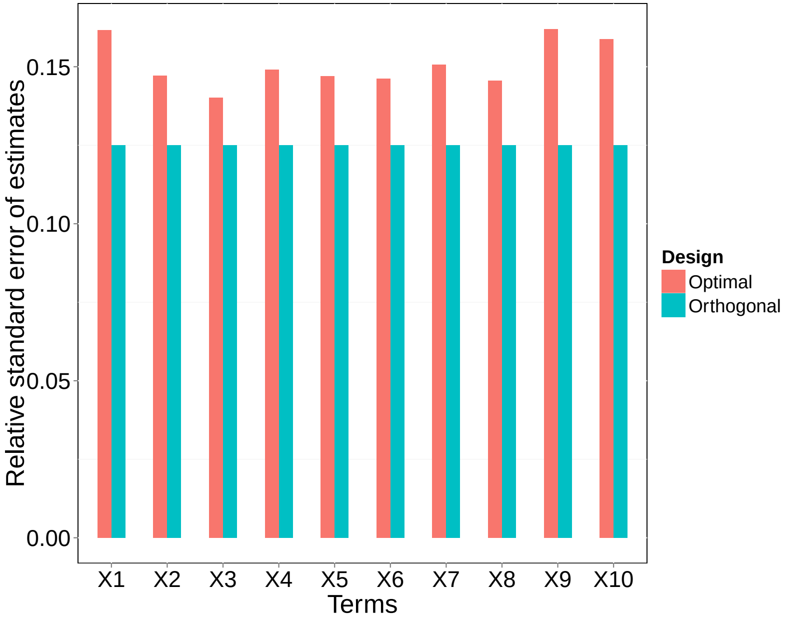
- A D-optimal design for a model containing the intercept, ME's and 2fi's (Atkinson et al., 2011)
- A D-optimal design with 64 runs and 10 factors was constructed using the R package **AlgDesign** (Wheeler, 2011)

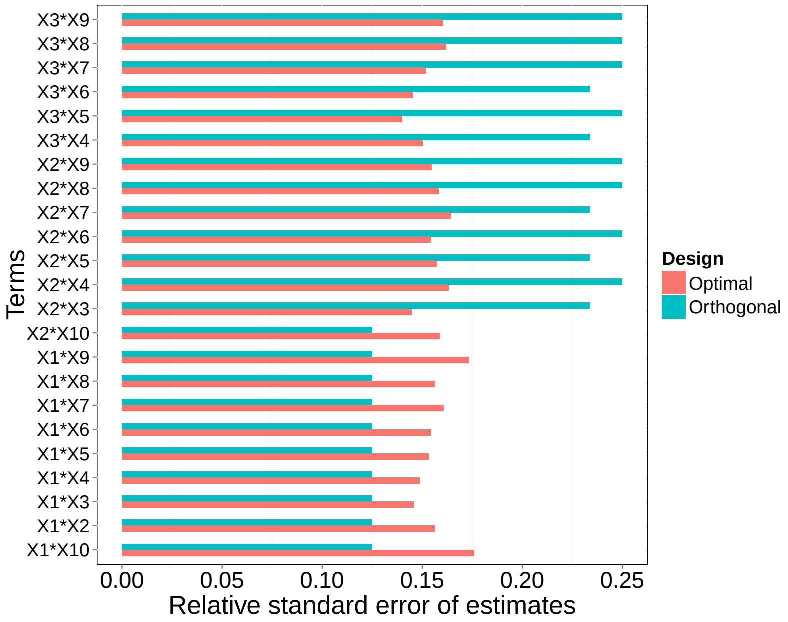
Relative D-efficiency of the orthogonal design **64.10** to the D-optimal design.

$$\text{Rel. D}_{eff} = \left(\frac{|X'_{orth} X_{orth}|}{|X'_d X_d|} \right)^{1/p} = 0.95$$

$p = 56$ parameters in the model

Standard error of estimates





Terms
X3*X9
X3*X8
X3*X7
X3*X6
X3*X5
X3*X4
X2*X9
X2*X8
X2*X7
X2*X6
X2*X5
X2*X4
X2*X3
X2*X10
X1*X9
X1*X8
X1*X7
X1*X6
X1*X5
X1*X4
X1*X3
X1*X2
X1*X10
0.00
0.05
0.10
0.15
0.20
0.25
Relative standard error of estimates
Design
Optimal
Orthogonal

Conclusions

This methodology can be used to construct nonregular designs with 64, 80, 96, 112, and 128 runs, and up to 33 factors.

VNS algorithm has been proven to be a fast and reliable alternative to construct combined designs.

The extra orthogonal column $\left[-\mathbf{1}^\top, \mathbf{1}^\top\right]^\top$ can be used to block D.

We can combine other orthogonal arrays such as strength-2 OAs, mixed-level OAs, etc.

Conclusions

This methodology can be used to construct nonregular designs with 64, 80, 96, 112, and 128 runs, and up to 33 factors.

VNS algorithm has been proven to be a fast and reliable alternative to construct combined designs.

The extra orthogonal column $\left[-\mathbf{1}^\top, \mathbf{1}^\top\right]^\top$ can be used to block D.

We can combine other orthogonal arrays such as strength-2 OAs, mixed-level OAs, etc.

Questions?

Thank you!

References

Atkinson, A.C., Donev, A.N. & Tobias R. (2007). *Optimum Experimental Designs, with SAS*. Oxford: Clarendon Press.

Chen, J., Sun, D.X. & Wu, C.F.J. (1999). A catalogue of two-level and three-level fractional factorial designs with small runs. *Internat. Statist. Rev.*, 61, p.p. 131-145.

Deng, L.-Y. & Tang, B. (1999). Generalized resolution and minimum aberration criteria for Plackett-Burman and other nonregular factorial design. *Statistica Sinica*, 9, p.p. 1071-1082.

Grömping, U. (2014). **FrF2***: Fractional Factorial Designs with 2-Level Factors*. R package version 1.6-9, URL `http://CRAN.R-project.org/package=FrF2`.

Li, W. & Lin, D.K.J. (2015). A note on foldover of 2^{k-p} designs with column permutations. To appear in *Technometrics*.

Mee, R. (2009). *A Comprehensive Guide to Factorial Two-Level Experimentation*. Springer.

Schoen, E.D. & Mee, R.W. (2012). Two-level designs of strength-3 and up to 48 runs. *Appl. Statist.* 61. Part 1, p.p. 163-174.

Schoen, E.D., Eendebak, P.T. & Nguyen, M.V.M. (2010). Complete enumeration of pure-level and mixed-level orthogonal arrays. *Journal of Combinatorial Designs*, 18(2), p.p. 123-140.

Tang, B. and Deng, L.-Y. (1999). Minimum G_2-aberration for nonregular fractional factorial designs. *The Annals of Statistics*, 27(6), p.p 1914-1926.

Wheeler, B. (2011). **AlgDesign***: Algorithmic Experimental Design*. R package version 1.1-7, URL `http://CRAN.R-project.org/package=AlgDesign`.

Xu, H. & Wong, A. (2007). Two-level nonregular designs from quaternary linear codes. *Statistica Sinica* 17, p.p. 1191-1213.