Fast Computation of Exact G-Optimal Designs Via I_λ-Optimality

Chris Nachtsheim
Carlson School of Management
University of Minnesota

Fall Technical Conference, West Palm Beach
October 4, 2018

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

Joint work with Lucia Hernandez:

Assistant Professor

School of Economics and Statistics

National University of Rosario

Rosario, Argentina

- **My former PhD student**
- **Carried out her dissertation research at Minnesota on Fulbright grant**

My Fulbright Trip to Argentina October, 2017

Studying off line quality control of wines in the Mendoza wine region with Lucia

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

And now---for those with suspicious minds in this time of the “Me Two” movement…

And now---for those with suspicious minds in this time of the “Me Two” movement…

My wife took that picture!

Here we are in the Andies near Chilean border

Same trip

Here we are in the Andies near Chilean border

My wife Maureen and two of our best friends

Here we are in the Andies near Chilean border

Lucia and her sister and parents

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

First, a little bit of optimal design theory

- **In our first submission, we skipped this**
- **Editor: add a primer!!!**

First, a little bit of optimal design theory

- **In our first submission, we skipped this**
- **Editor: add a primer!!!**

So here we go ...

Basic notation

- **Standard linear model:**

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- **X is *n* by *p*, β is p by 1, and the ith row of X is:**

$$\mathbf{f}'(\mathbf{x}_i)$$

- **So that the ith observation can be written:**

$$y_i = \mathbf{f}'(\mathbf{x}_i)\boldsymbol{\beta} + \epsilon_i$$

- **Assume constant error variance, and, WLOG: $\sigma^2 = 1$**

What is an approximate design?

- **An approximate design is a probability measure ξ on a design space χ**
- **Design problem: Quadratic regression on χ = [-1, 1], and we can take n = 16 runs**
- **Example design measure ξ:**

Place 1/3 weight at -1	**16/3 = 5.333 runs**
Place 1/3 weight at 0	**16/3 = 5.333 runs**
Place 1/3 weight at +1	**16/3 = 5.333 runs**

- **Can only implement "approximately."**

Information matrix of approximate design ξ

- **Information matrix:**

$$\mathbf{M}(\xi) = \int_{\chi} \mathbf{f}(\mathbf{x})\mathbf{f}'(\mathbf{x})d\xi(\mathbf{x})$$

- **By Caratheodory's theorem, any continuous measure can be discretized:**

$$\mathbf{M}(\xi) = \sum_{i=1}^{s} \mathbf{f}(\mathbf{x_i})\mathbf{f}'(\mathbf{x_i})\xi(\mathbf{x}_i)$$

- **So, we can focus on discrete probability measures**

Discretization example

- **Approximate design ξ is given by the uniform probability measure on χ = [-1, 1].**
- **Model is quadratic: $f^T(x) = (1, x, x^2)$**

$$\mathbf{M}(\xi) = \int_{\chi} \mathbf{f}(\mathbf{x})\mathbf{f}'(\mathbf{x})d\xi(\mathbf{x}) = \int_{-1}^{1} \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix} (1, x, x^2) f(x) dx$$

$$= \int_{-1}^{1} \begin{pmatrix} 1 & x & x^2 \\ x & x^2 & x^3 \\ x^2 & x^3 & x^4 \end{pmatrix} \frac{1}{2} dx = \begin{pmatrix} 1 & 0 & 1/3 \\ 0 & 1/3 & 0 \\ 1/3 & 0 & 1/5 \end{pmatrix}$$

- **The discrete design placing 5/18 weight at $\pm\sqrt{3/5}$ and 4/9 weight at 0 yields the same M**

What is an exact design?

- **Assume n a positive integer and for ξ a discrete design measure on χ.**
- **ξ a an exact n-point design if nξ(x) is a positive integer** $\forall \mathbf{x} \in \chi$
- **Example: n = 3 (or 6 or 9 or 12 …)**

x:	-1	0	1
$\xi_3(x)$:	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

M matrix for exact design and some notation

- Assume n (not necessarily distinct) points $x_1, \ldots x_n$.
- Information matrix is:

$$\mathbf{M}(\xi_n) = \sum_{i=1}^{n} \mathbf{f}(\mathbf{x}_i)\mathbf{f}'(\mathbf{x}_i)\frac{1}{n} = \mathbf{X}'\mathbf{X}/n$$

- Notation: Sets of possible designs

Ξ = set of all approximate designs

Ξ_n = set of all n-point exact designs

D-Optimal designs

Definition: The D-optimal design maximizes the determinant of the information matrix:

$$\xi^D = \arg\max_{\xi \in \Xi} |\mathbf{M}(\xi)| \qquad \textbf{(Approximate case)}$$

$$\xi_n^D = \arg\max_{\xi_n \in \Xi_n} |\mathbf{M}(\xi_n)| \qquad \textbf{(Exact case)}$$

Why D?

- **D-optimal designs minimize the volume of the confidence region for β.**
- **D-optimal designs best for estimation of β.**

D-optimal approximate design

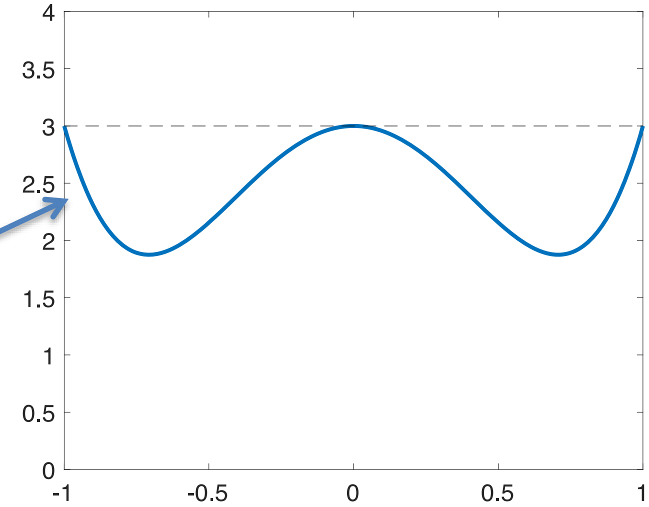
$$\xi^D = \arg\max_{\xi \in \Xi} |\mathbf{M}(\xi)|$$

Example: Quadratic regression on χ = [-1,1]

$$\xi^D = \left\{ \begin{array}{ccc} -1 & 0 & 1 \\ 1/3 & 1/3 & 1/3 \end{array} \right\}$$

|M| = 4/27

Variance of prediction



What if prediction is of primary interest?

- **Normalized variance of prediction:**

$$v(\mathbf{x},\xi) = \begin{cases} \mathbf{f}'(\mathbf{x})\mathbf{M}^{-1}(\xi)\mathbf{f}(\mathbf{x}) & \text{if } \mathbf{M}(\xi) \text{ is not singular} \\ \infty & \text{if } \mathbf{M}(\xi) \text{ is singular} \end{cases}$$

- **Pick the design to minimize the maximum variance of prediction (G-optimality)**
- **Pick the design to minimize the average variance of prediction (I-optimality)**

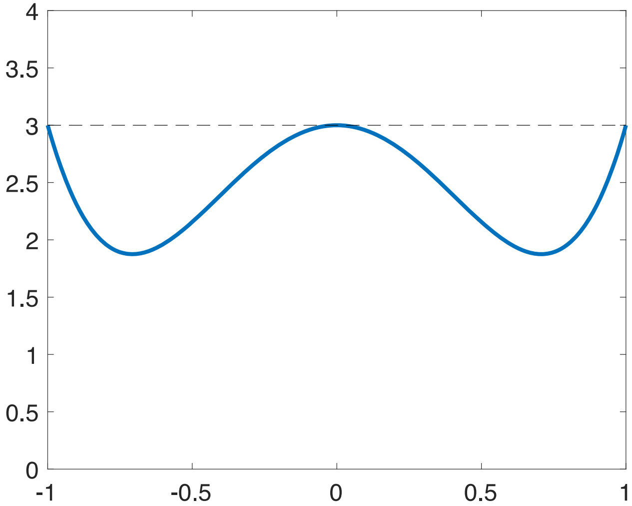
G-optimal (minimax) design

$$\xi^G = \arg\min_{\xi \in \Xi} \max_{\mathbf{x} \in \chi} v(\mathbf{x}, \xi)$$

Example: Quadratic regression on χ = [-1,1]

$$\xi^G = \left\{ \begin{array}{ccc} -1 & 0 & 1 \\ 1/3 & 1/3 & 1/3 \end{array} \right\}$$

Minimax v(x) = 3
Average v(x) = 2.4



Wait! That G-optimal design looked just like the D-optimal design

$$\xi^D = \begin{Bmatrix} -1 & 0 & 1 \\ 1/3 & 1/3 & 1/3 \end{Bmatrix}$$

$$\xi^G = \begin{Bmatrix} -1 & 0 & 1 \\ 1/3 & 1/3 & 1/3 \end{Bmatrix}$$

Kiefer-Wolfowitz Equivalence Theorem (1959)

The following conditions are equivalent:

1. **ξ is D-optimal.**
2. **ξ is G-optimal.**
3. $\max_{\mathbf{x} \in \chi} v(\mathbf{x}, \xi) = p$

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

K-W does NOT hold for exact designs

- **D-optimal and G-optimal designs are not necessarily the same for finite n**
- **Fast algorithms exist for D-optimal designs**
- **Fast algorithms do not exist for G-optimal designs**

OK, so why is G-optimality hard?

D-optimal exchange algorithms

1. Generate a random starting design
2. Cycle through the n points in the design:
   – For point i, find the point in the the design space x*, such that when x* is exchanged for x_i, we get a maximal increase in the determinant.
3. Repeat step 2 until no further improvements

An optimization over c is required for every design point -- function to be evaluated is the determinant

OK, so why is G-optimality hard?

G-optimal exchange algorithms

1. Generate a random starting design
2. Cycle through the n points in the design:
   - For point i, find the point in the the design space x*, such that when x* is exchanged for x_i, we get a maximal decrease in the maximum variance.
3. Repeat step 2 until no further improvements

An optimization over c is required for every design point – but to evaluate the criterion must do another maximization of the variance function

Now, let's beat a dead dog:

Coordinate exchange is really bad with G-optimality.

1) **The criterion is the maximum variance over the design space. Algorithms that attempt to do minimax are very expensive!!!!**

2) **The algorithm fails regularly to find the global optimum.**

Example: Quadratic regression, n = 6

We know the G-optimal design places two observations at the each of the endpoints and two at the center.

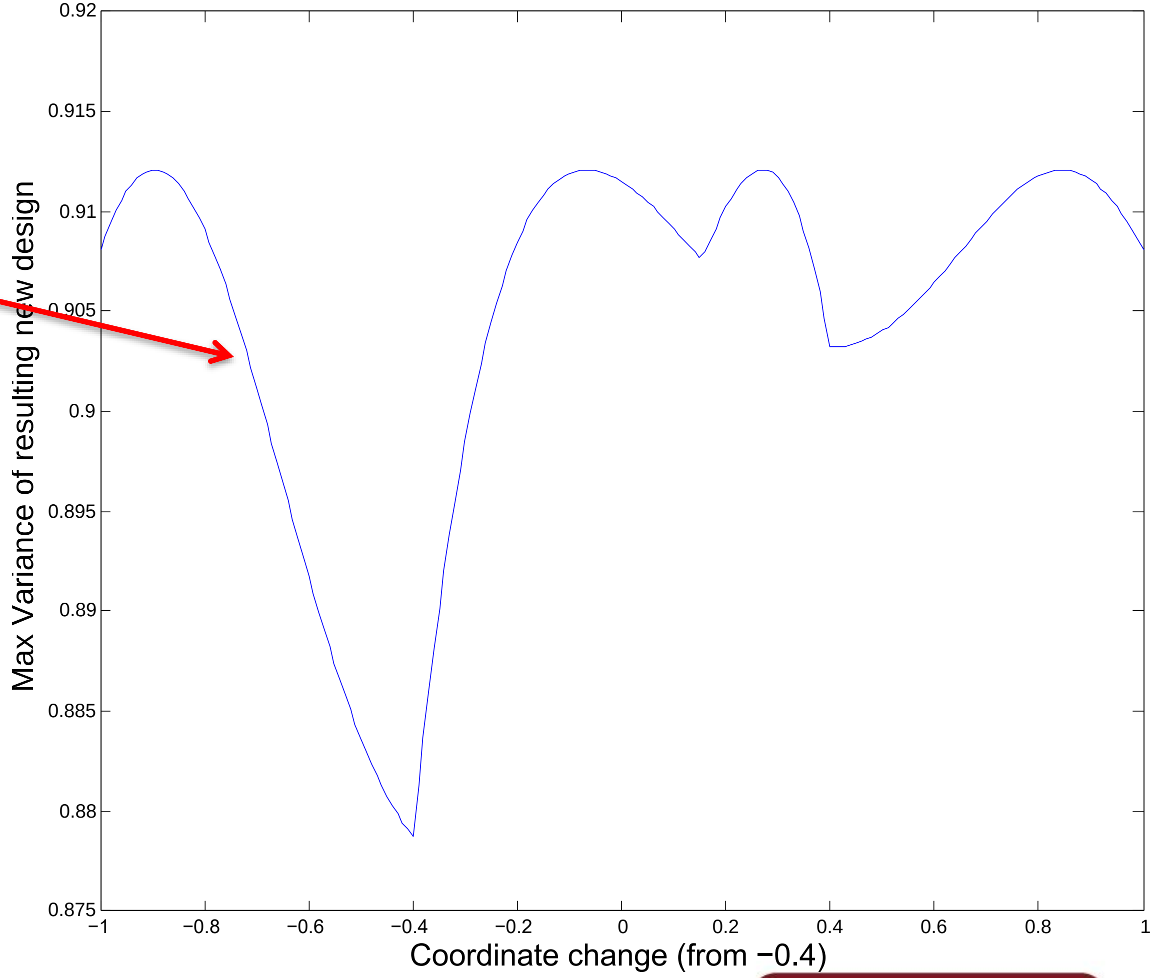
Consider this starting design:

-1 -.4 0 0 .4 1

Seems clear we want to move .4 to 1 and -.4 to -1.

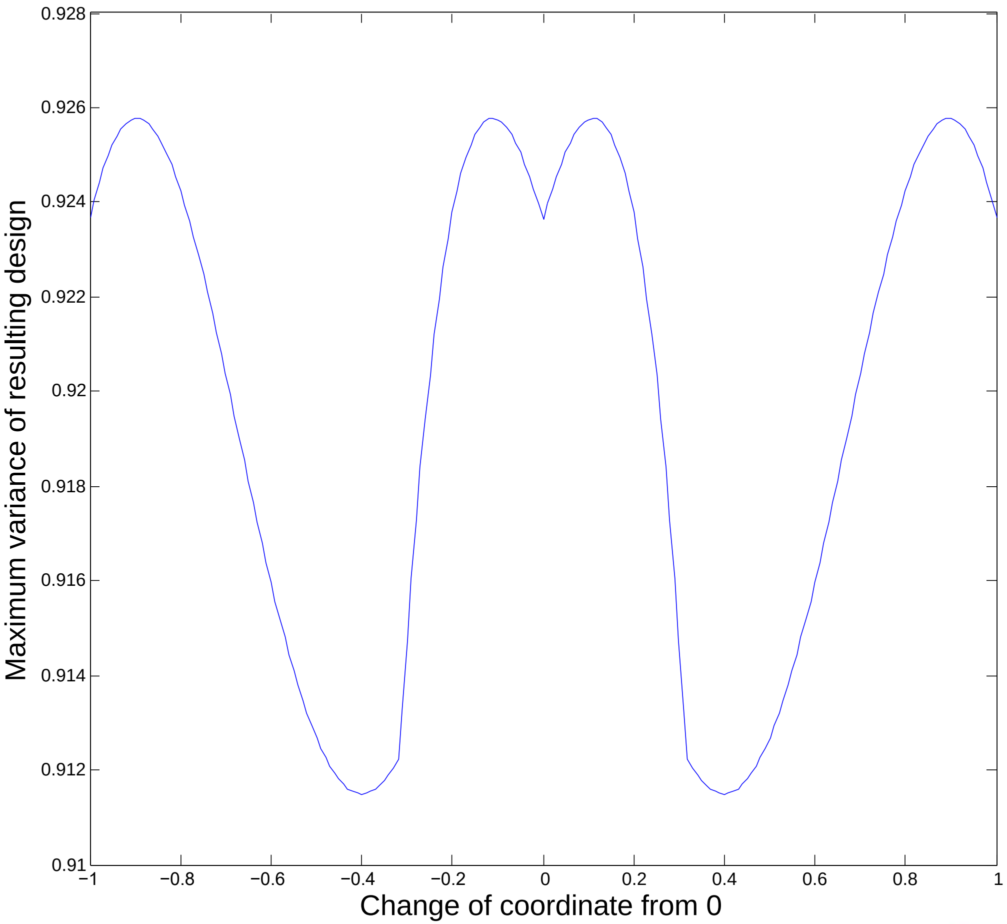
When you try to change the -.4 coordinate, you can't

Resulting maximum variance from exchange of -0.4



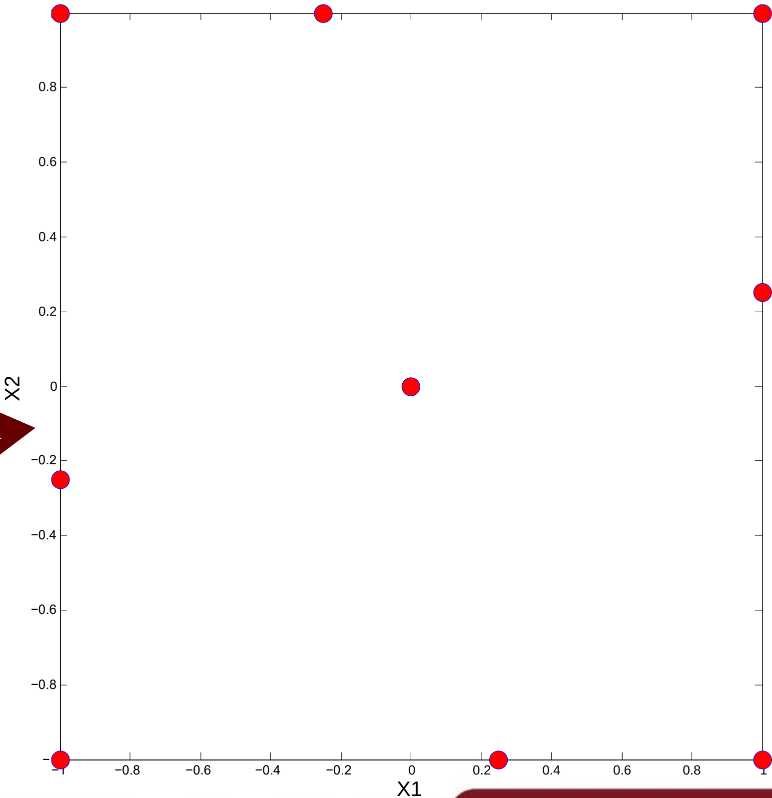
CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

When you try to change the 0 coordinate, your best move is to change it to +/- 0.4…and then you're stuck



G-optimal algorithm success rates are bad using random starting designs

1. One factor, n = 6, 86% success rate
2. Two factors, n = 9, 0% success rate. Best design



CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

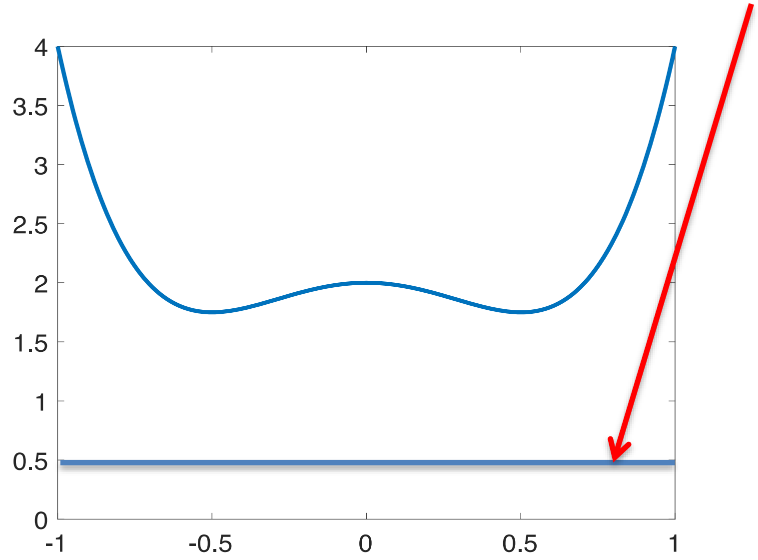
I_λ-optimal (minimum average variance) Design

$$\xi^{I_\lambda} = \arg\min_{\xi \in \Xi} \int_{\chi} v(\mathbf{x}, \xi)\lambda(\mathbf{x})d\mathbf{x}$$

Example: Quadratic regression on χ = [-1,1]; λ(x) uniform

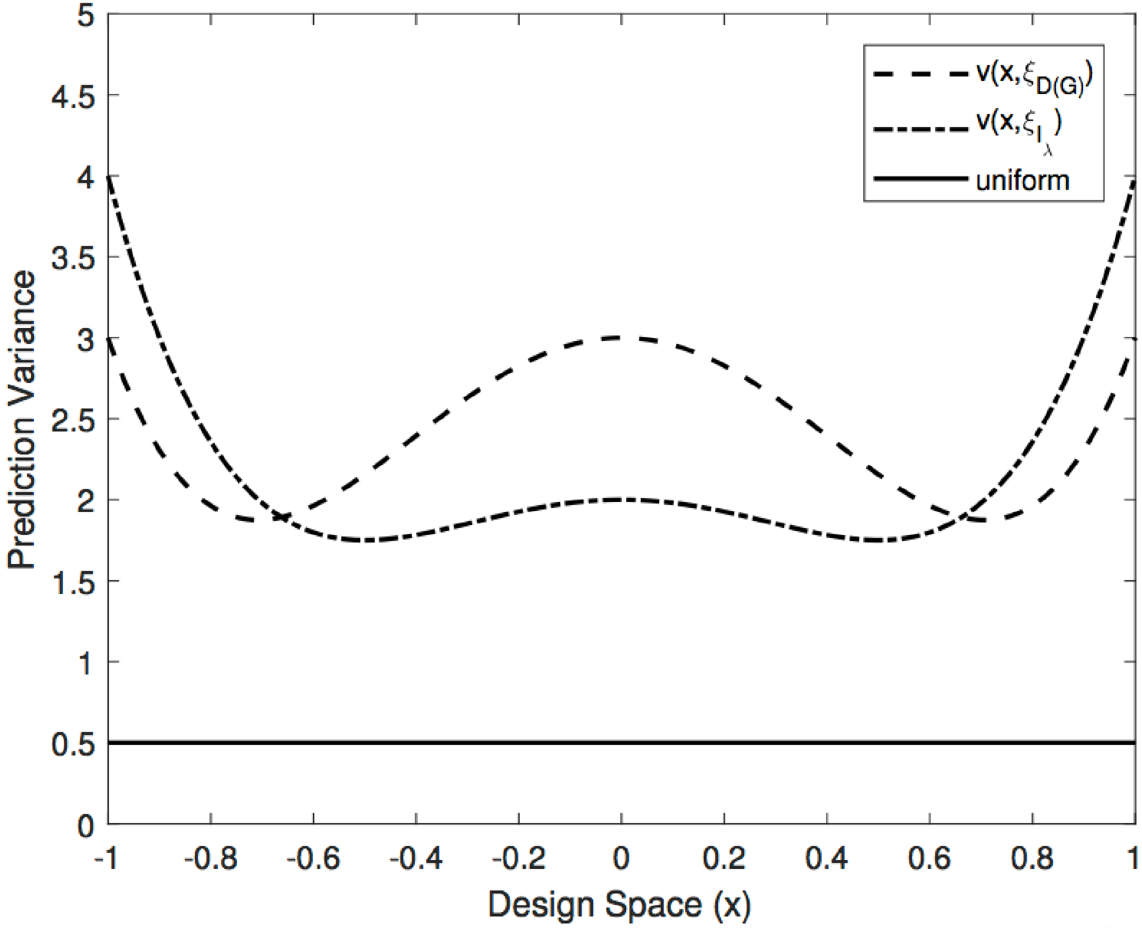
$$\xi^{I_\lambda} = \left\{ \begin{array}{ccc} -1 & 0 & 1 \\ 1/4 & 1/2 & 1/4 \end{array} \right\}$$

Minimax v(x) = 4
Average v(x) = 2.133



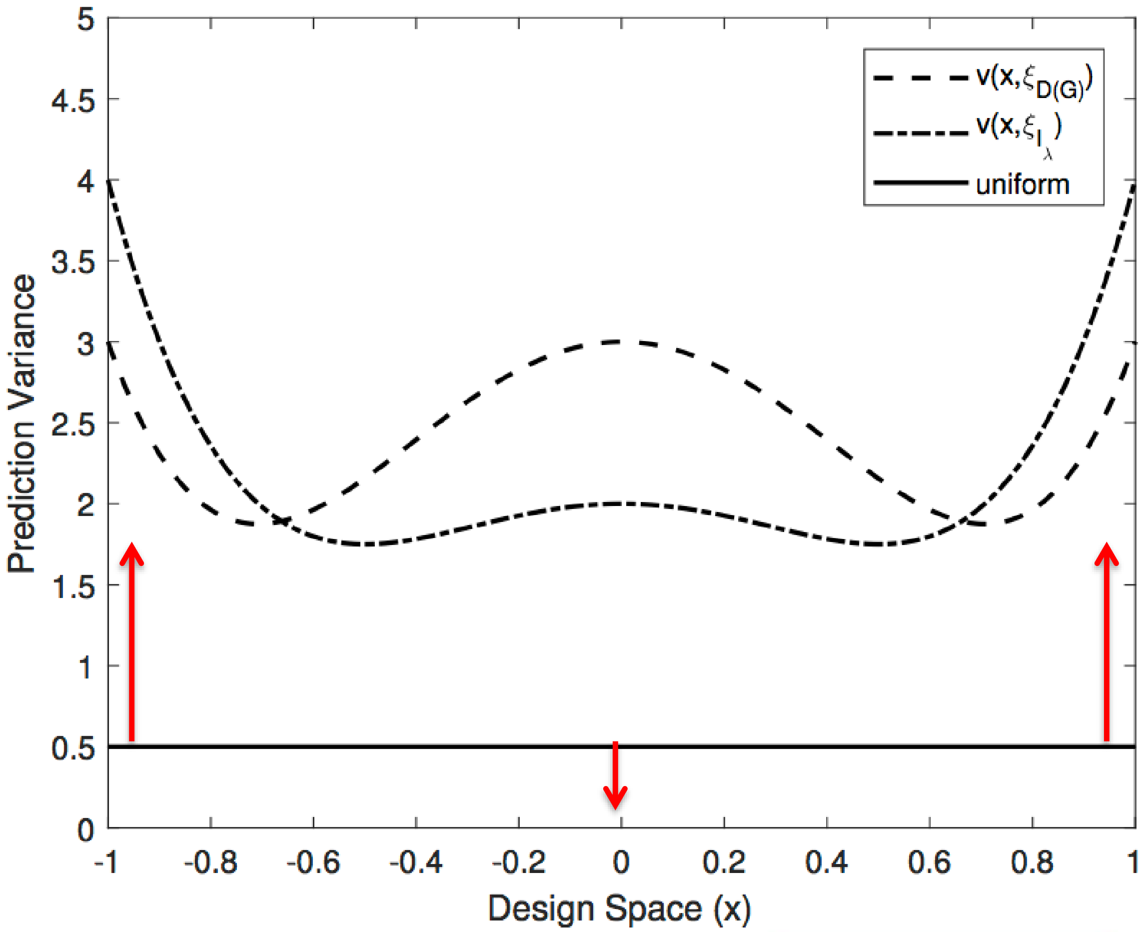
Superimpose G and I_λ optimal variance functions

What if we increased $\lambda(x)$ near the boundaries and reduced it in the center?



Superimpose G and I_λ optimal variance functions

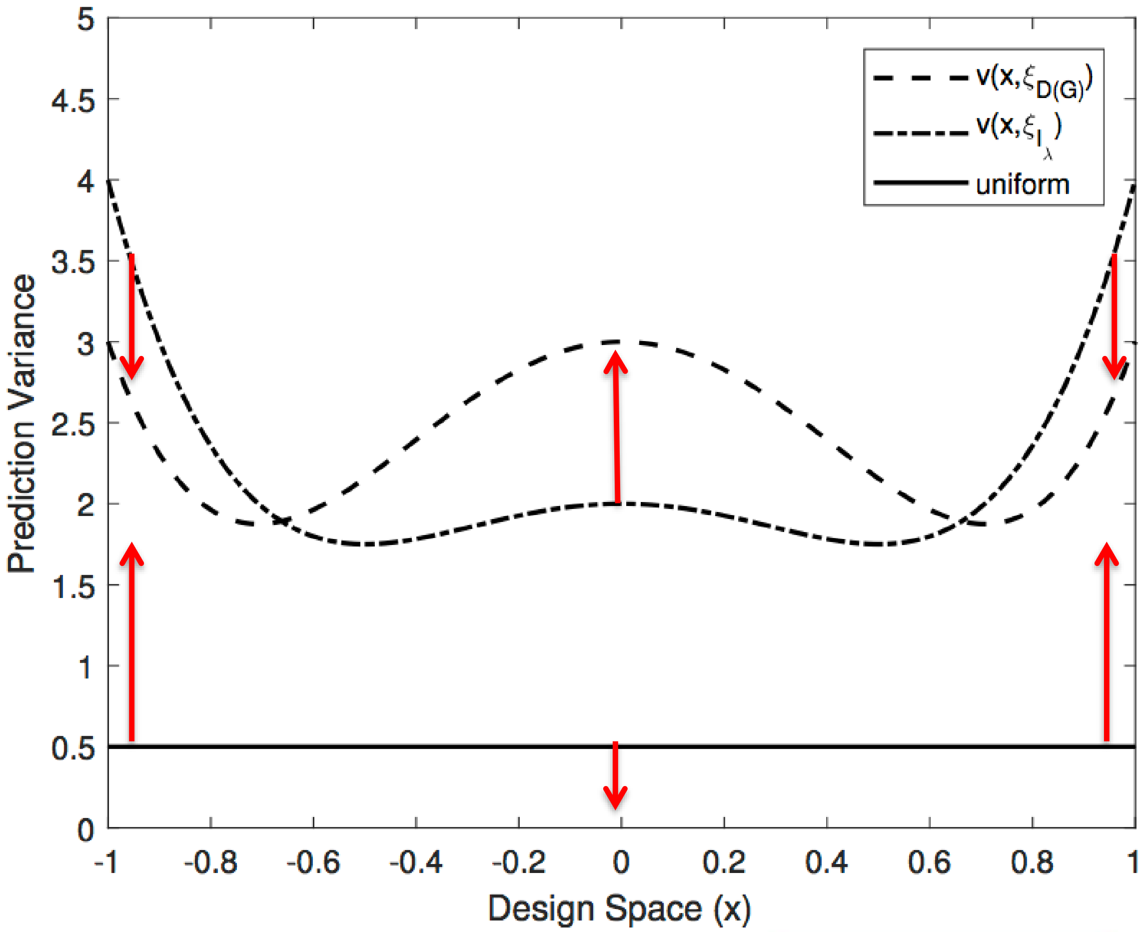
What if we increased λ(x) near the boundaries and reduced it in the center?



Superimpose G and I_λ optimal variance functions

What if we increased λ(x) near the boundaries and reduced it in the center?

That should bring down the maximum variance of prediction at the boundaries



CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

Research Question

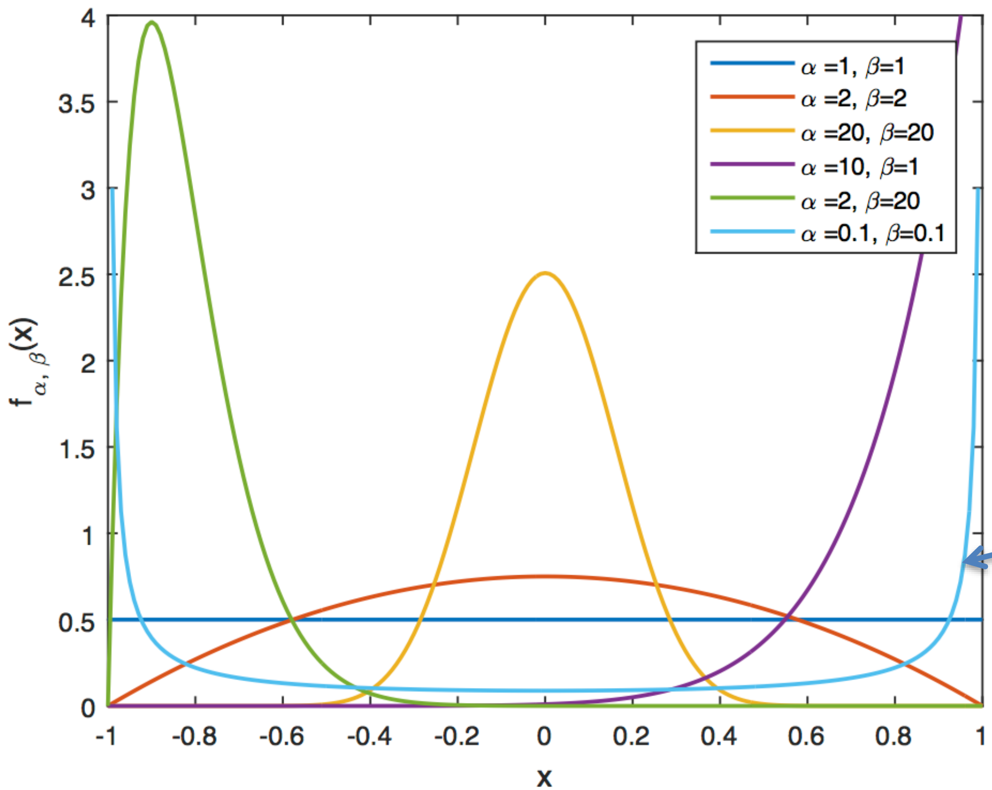
Can a clever choice of the weight function λ yield:

$$\xi^{G(D)} \approx \xi^{I_\lambda}$$

If so, we can use standard, fast exchange algorithms (with no minimax search) to find the G-optimal design

Try λ(x) = beta density on [-1,1]

$$f_{\alpha,\beta} = \left(\frac{1}{2}\right)^{\alpha+\beta-1} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(1+x)^{\alpha-1}(1-x)^{\beta-1} \quad -1 \le x \le 1 \;\; \alpha,\beta > 0$$

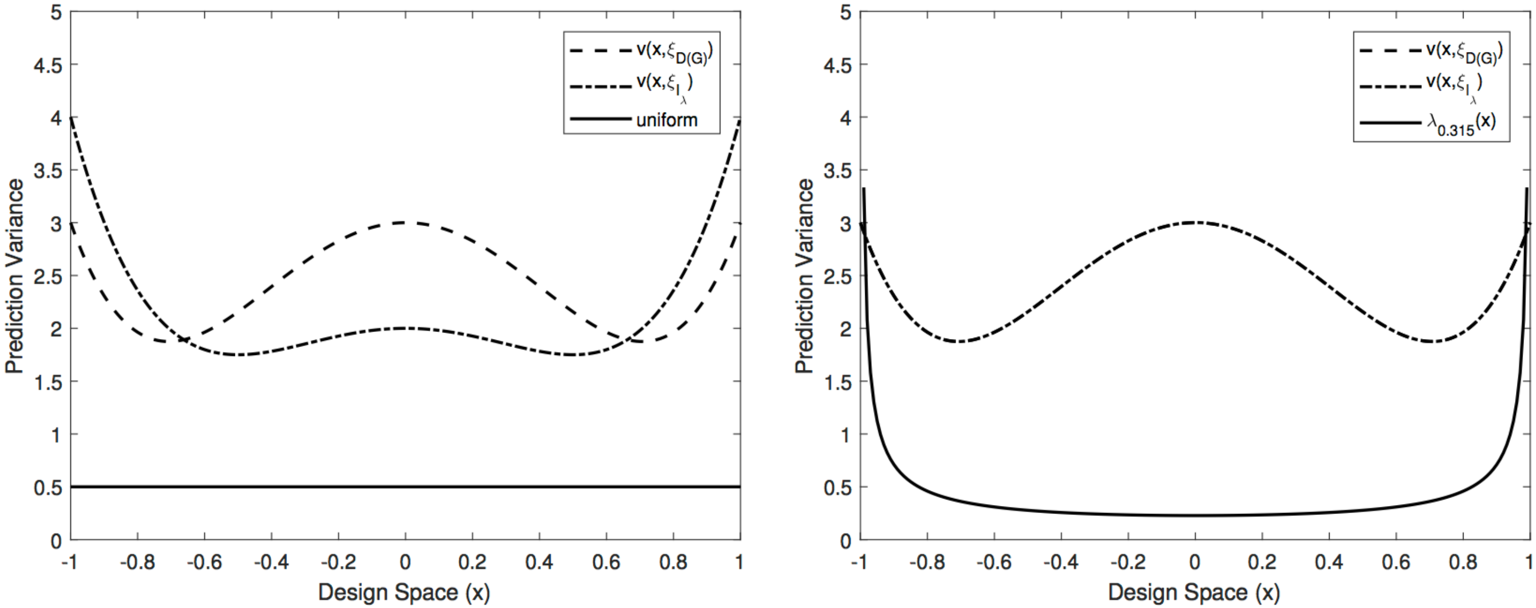


Blue guy has α = β < 1, bathtub shape—what we want

Find α = β so that designs are the same

$$\alpha^* = \arg\min_{\alpha} \int_{\chi} [d(x, \xi^{G(D)}) - d(x, \xi^{I_\lambda})]^2 d\lambda_\alpha(x)$$

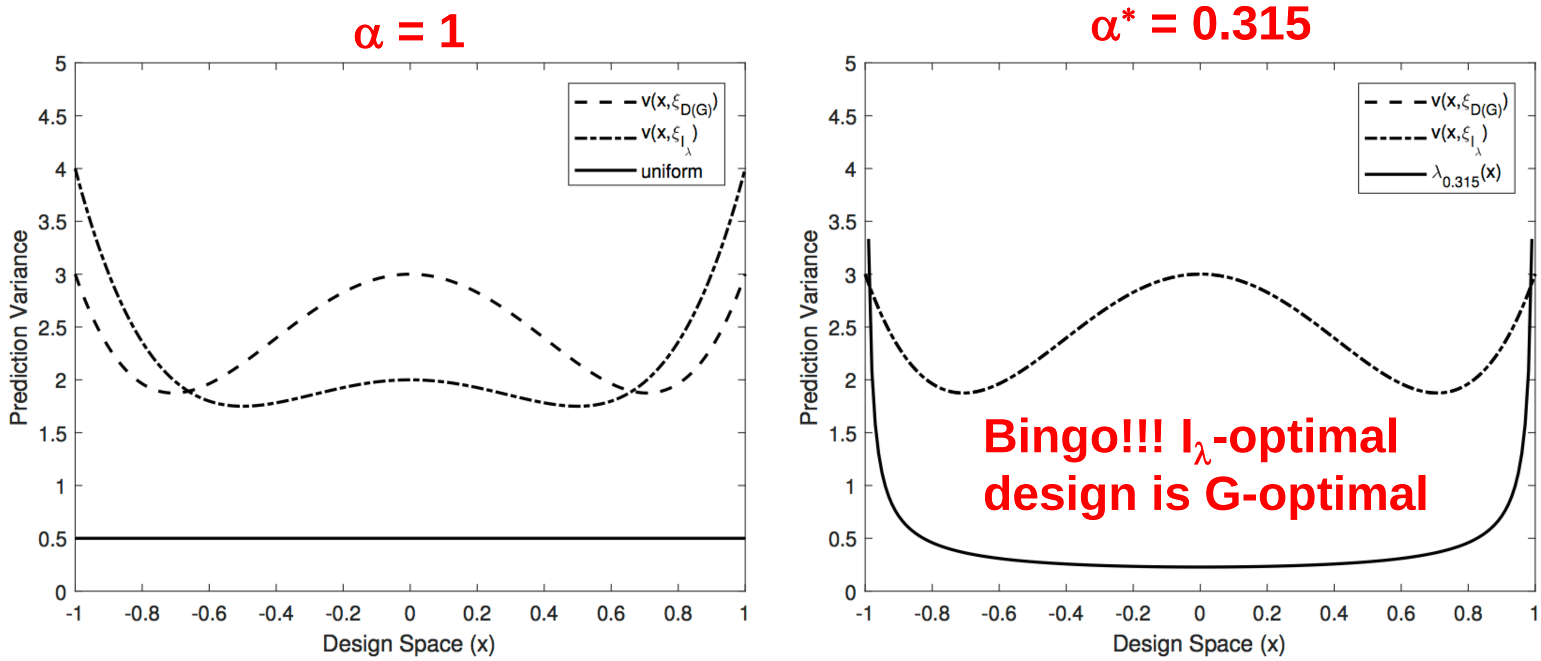
α = 1 **α* = 0.315**



(a) Uniform weight function (λ) (b) Bathtub-shaped weight function (λ)

Find α = β so that designs are the same

$$\alpha^* = \arg\min_{\alpha} \int_{\chi} [d(x, \xi^{G(D)}) - d(x, \xi^{I_\lambda})]^2 d\lambda_\alpha(x)$$



(a) Uniform weight function (λ) (b) Bathtub-shaped weight function (λ)

OK, how about more factors

- **Need multidimensional gamma density for weight function**
- **Assume independence, product of gammas?**

OK, how about more factors

- **Need multidimensional gamma density for weight function**
- **Assume independence, product of gammas?**

Failure! Dead end.
Abandon ship..

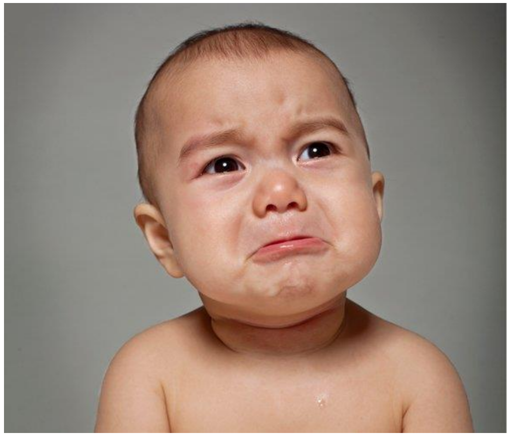
CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

OK, how about more factors

- **Need multidimensional gamma density for weight function**
- **Assume independence, product of gammas?**

Failure! Dead end.
Abandon ship..

Lucia was sad



CARLSON
SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

But wait!

- **Think about the I_λ criterion:**

$$\int_{\chi} v(\mathbf{x},\xi)\lambda(\mathbf{x})d\mathbf{x} = \mathrm{tr}[\mathbf{M}^{-1}(\xi)\int_{\chi}\mathbf{f}'(\mathbf{x})\mathbf{f}(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}] = \mathrm{tr}[\mathbf{M}^{-1}(\xi)\mathbf{W}]$$

where:

$$\mathbf{W} = \int_{\chi}\mathbf{f}'(\mathbf{x})\mathbf{f}(\mathbf{x})\lambda(\mathbf{x})d\mathbf{x}$$

- **Only purpose of the λ density is to produce W**
- **So skip choosing λ – try choosing W directly**

But W has p(p+1)/2 entries---daunting?

$$\mathbf{W} = \int_{\chi} \mathbf{f}(x)\mathbf{f}'(x)d\lambda_{\alpha}(x) = \int_{\chi} \begin{pmatrix} 1 & x & x^2 \\ x & x^2 & x^3 \\ x^2 & x^3 & x^4 \end{pmatrix} \lambda_{\alpha}(x)d(x)$$

$$= \begin{pmatrix} 1 & E_{\lambda}(x) & E_{\lambda}(x^2) \\ E_{\lambda}(x) & E_{\lambda}(x^2) & E_{\lambda}(x^3) \\ E_{\lambda}(x^2) & E_{\lambda}(x^3) & E_{\lambda}(x^4) \end{pmatrix} = \begin{pmatrix} 1 & w_1 & w_2 \\ w_1 & w_2 & w_3 \\ w_2 & w_3 & w_4 \end{pmatrix}$$

But W has p(p+1)/2 - 1 entries---daunting?

$$\mathbf{W} = \int_{\chi} \mathbf{f}(x)\mathbf{f}'(x)d\lambda_\alpha(x) = \int_{\chi} \begin{pmatrix} 1 & x & x^2 \\ x & x^2 & x^3 \\ x^2 & x^3 & x^4 \end{pmatrix} \lambda_\alpha(x)d(x)$$

$$= \begin{pmatrix} 1 & E_\lambda(x) & E_\lambda(x^2) \\ E_\lambda(x) & E_\lambda(x^2) & E_\lambda(x^3) \\ E_\lambda(x^2) & E_\lambda(x^3) & E_\lambda(x^4) \end{pmatrix} = \begin{pmatrix} 1 & w_1 & w_2 \\ w_1 & w_2 & w_3 \\ w_2 & w_3 & w_4 \end{pmatrix}$$

But W has p(p+1)/2 - 1 entries---daunting?

$$\mathbf{W} = \int_{\mathcal{X}} \mathbf{f}(x)\mathbf{f}'(x)d\lambda_\alpha(x) = \int_{\mathcal{X}} \begin{pmatrix} 1 & x & x^2 \\ x & x^2 & x^3 \\ x^2 & x^3 & x^4 \end{pmatrix} \lambda_\alpha(x)d(x)$$

$$= \begin{pmatrix} 1 & E_\lambda(x) & E_\lambda(x^2) \\ E_\lambda(x) & E_\lambda(x^2) & E_\lambda(x^3) \\ E_\lambda(x^2) & E_\lambda(x^3) & E_\lambda(x^4) \end{pmatrix} = \begin{pmatrix} 1 & w_1 & w_2 \\ w_1 & w_2 & w_3 \\ w_2 & w_3 & w_4 \end{pmatrix}$$

All we know is that $-1 \le w_i \le 1$

W-entries for full quadratic—Yes, daunting

Factors	p	W-entries
1	3	5
2	6	20
3	10	54
4	15	119
5	21	230
6	28	405

Why is this a problem?

I need to find the W matrix that leads the I_λ-optimality (coord exch) algorithm to find the minimax design

1. **Fix W**
2. **Now use coordinate exchange to find $\xi_n|W$.**
   - **This is (pretty) FAST**
3. **OK, now evaluate $\xi_n|W$ using the minimax criterion**
   - **This is SLOW**
4. **Change W (move toward optimality), go to 2.**

This is a minimax algorithm in W

Why is this a problem?

I need to find the W matrix that leads the I_λ-optimality (coord exch) algorithm to find the minimax design

1. **Fix W**
2. **Now use coordinate exchange to find $\xi_n|W$.**
   - **This is (pretty) FAST**
3. **OK, now evaluate $\xi_n|W$ using the minimax criterion**
   - **This is SLOW**
4. **Change W (move toward optimality), go to 2.**

NOOOOOO!!

Is there a way to guess at W* in advance?

Little Theorem (Nachtsheim, 1979).

Let $\xi^{G(D)}$ denote the G(D)-optimal design for model f in design space χ, and let ξ^{λ} denote the I_{λ}-optimal design. Then:

$$\xi^{G(D)} = \lambda \Rightarrow \xi^{\lambda} = \xi^{G(D)}$$

Is there a way to guess at W* in advance?

Little Theorem (Nachtsheim, 1979).

Let $\xi^{G(D)}$ denote the G(D)-optimal design for model f in design space χ, and let ξ^{λ} denote the I_{λ}-optimal design. Then:

$$\xi^{G(D)} = \lambda \Rightarrow \xi^{\lambda} = \xi^{G(D)}$$

Translated: Use the approximate G(D)-optimal design as the weight function λ. Then the I_l-optimal design is the G(D)-optimal design

Implication:

Really good guess for the optimal W:

$$\mathbf{W}^* \approx \int_{\chi} \mathbf{f}(\mathbf{x})\mathbf{f}^T(\mathbf{x})d\xi^{G(D)}(\mathbf{x})$$

That is: a really good guess at the best W, is the information matrix for the G(D)-optimal design!

OK, we have a starting value for W

- **But we still have the dimensionality problem, no?**
- **Well, turns out that goes away too!**
- **Why? The information matrix for the G(D)-optimal designs for full second-order models have only two unique values, w_1 and w_2**
- **Upshot: Dimensionality in W is 2!**

Example: Two-factor RSM model

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 0 & \boxed{0.744 \quad 0.744} \\ 0 & \boxed{0.744} & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{0.744} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.583 & 0 & 0 \\ 0.744 & 0 & 0 & 0 & \boxed{0.744} & 0.583 \\ 0.744 & 0 & 0 & 0 & 0.583 & \boxed{0.744} \end{pmatrix}$$

Example: Two-factor RSM model

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0.744 & 0.744 \\ 0 & 0.744 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.744 & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{0.583} & 0 & 0 \\ 0.744 & 0 & 0 & 0 & 0.744 & \boxed{0.583} \\ 0.744 & 0 & 0 & 0 & 0.583 & 0.744 \end{pmatrix}$$

Example: Two-factor RSM model

$$\mathbf{W} = \begin{pmatrix} 1 & 0 & 0 & 0 & \boxed{0.744} & \boxed{0.744} \\ 0 & \boxed{0.744} & 0 & 0 & 0 & 0 \\ 0 & 0 & \boxed{0.744} & 0 & 0 & 0 \\ 0 & 0 & 0 & \boxed{0.583} & 0 & 0 \\ 0.744 & 0 & 0 & 0 & \boxed{0.744} & \boxed{0.583} \\ 0.744 & 0 & 0 & 0 & 0.583 & \boxed{0.744} \end{pmatrix}$$

Amazingly, this is always true

$$W_{ij} = \begin{cases} 1: & i = j = 1 \\ w_1: & i = j = 2, \ldots, m+1;\ i = j = m(m+1)/2+2, \ldots, p; \\ & i = 1,\ j = m(m+1)/2+2, ..., p;\ j = 1,\ i = m(m+1)/2+2, ..., p; \\ w_2: & i = j = m+2, \ldots, m(m+1)/2+1; \\ & i = m(m+1)/2+2, \ldots, p,\ j = m(m+1)/2+2, \ldots, p,\ \text{and } i \neq j; \\ 0: & \text{elsewhere} \end{cases}$$

From Atkinson, Donev, Tobias Optimal Design Book (2007)

	Approximate G(D)-optimal design point weights				Non-zero **W** entries	
m	Center point	Edge Centers	Corner Points	s	w_1	w_2
1	0.333 (1)	-	0.333 (2)	3	0.667	0.667
2	0.096 (1)	0.080 (4)	0.146 (4)	9	0.744	0.583
3	0.066 (1)	0.035 (12)	0.064 (8)	21	0.793	0.651
4	0.047 (1)	0.016 (32)	0.028 (16)	49	0.828	0.702
5	0.036 (1)	0.007 (80)	0.013 (32)	113	0.852	0.739

Amazingly, this is always true

$$W_{ij} = \begin{cases} 1: & i = j = 1 \\ w_1: & i = j = 2, \ldots, m+1;\ i = j = m(m+1)/2+2, \ldots, p; \\ & i = 1,\ j = m(m+1)/2+2, ..., p;\ j = 1,\ i = m(m+1)/2+2, ..., p; \\ w_2: & i = j = m+2, \ldots, m(m+1)/2+1; \\ & i = m(m+1)/2+2, \ldots, p,\ j = m(m+1)/2+2, \ldots, p, \text{ and } i \neq j; \\ 0: & \text{elsewhere} \end{cases}$$

- **At least for number of factors up to five. We haven't looked further**

Finally, the algorithm

1. Obtain the approximate G(D)-optimal design, its information matrix, and the two starting w values
2. Use the coordinate exchange algorithm to find the optimal w_1 and w_2 values in a neighborhood of the starting values
   1. To evaluate w_1 and w_2, obtain the I_λ-optimal design given W and evaluate the maximum variance
   2. If the new maxvar is less than the best maxvar found, save the new maxvar and the new w_1 and w_2 values, and continue with the coordinate exchange algorithm on the w values until convergence

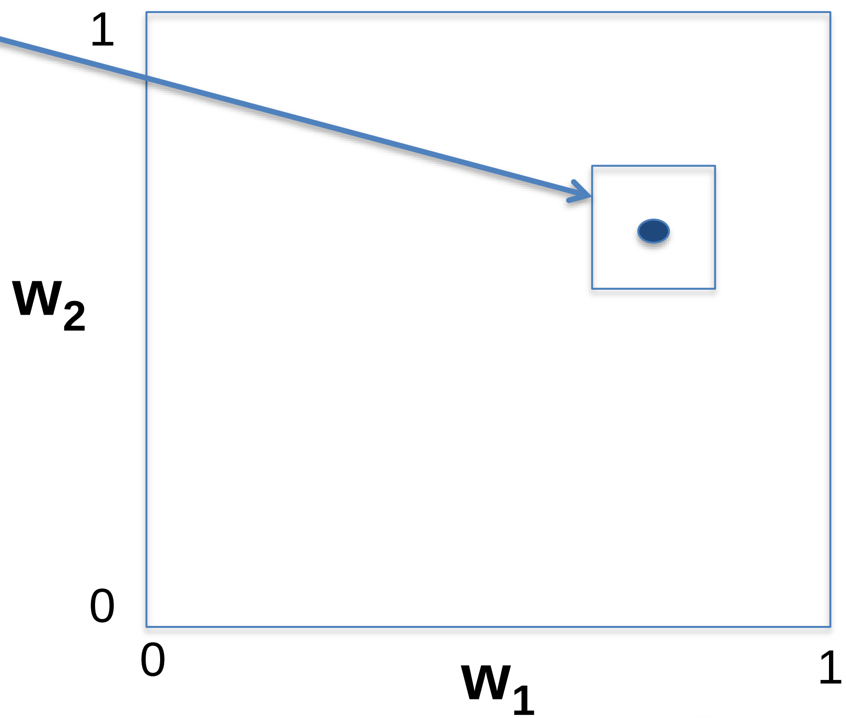
The "neighborhood" of w_1 and w_2 values

$$\mathcal{W} = [w_1^{\mathrm{G(D)}} - \Delta, w_1^{\mathrm{G(D)}} + \Delta] \times [w_2^{\mathrm{G(D)}} - \Delta, w_2^{\mathrm{G(D)}} + \Delta]$$

The “neighborhood” of w_1 and w_2 values

$$\mathcal{W} = [w_1^{\mathrm{G(D)}} - \Delta, w_1^{\mathrm{G(D)}} + \Delta] \times [w_2^{\mathrm{G(D)}} - \Delta, w_2^{\mathrm{G(D)}} + \Delta]$$

- **We use $\Delta = 0.1$**



Performance

We have to do a 2D minimax search over a small space

- **We might predict that the standard G-optimal algorithm based on the coordinate exchange will do better in one and two dimensions.**
- **But for three or more factors, this algorithm should be better**

Competitors (But only consider 1 – 3 factors)

1. **Genetic programming**
   - Borkowski, J. J. (2003). "Using a genetic algorithm to generate small exact response surface designs". *Journal of Probability and Statistical Science* 1(1), 65–88.
2. **Standard minimax coordinate exchange algorithm**
   - Rodriguez, M., Jones, B., Borror, C. M. and Montgomery, D. C. (2010). "Generating and assessing exact G-optimal designs." *Journal of Quality Technology* 42(1), 1–18.

Borkowski (2003): Genetic Algorithm (GA)

1. **John Borkowski's paper was first, and well done.**
2. **Identified the following test problems**

Factors	Run sizes (n)						
1	3	4	5	6	7	8	9
2	6	7	8	9	10	11	12
3	10	11	12	13	14	15	16

3. **Found some very good designs (tough to beat)**

Borkowski (2003): Genetic Algorithm (GA)

1. **John Borkowski's paper was first, and well done.**
2. **Identified the following test problems**

Factors	Run sizes (n)						
1	3	4	5	6	7	8	9
2	6	7	8	9	10	11	12
3	10	11	12	13	14	15	16

3. **Found some very good designs (tough to beat)**
4. **Takes forever!**

Rodriguez, Jones, Borror, Montgomery

- Used standard coordinate exchange where the objective is to minimize the maximum variance
- Used same test problems
- Mixed results – but did find a design or two that was better than the GP designs
- Faster than GP

Comparisons: Quality of Design

Efficiencies of our designs relative to G-CEXCH and GA

One factor			Two factors			Three factors		
n	G-CEXCH	GA	n	G-CEXCH	GA	n	G-CEXCH	GA
3	100.0%	100.0%	6	97.5%	96.1%	10	102.0%	95.4%
4	97.4%	96.2%	7	97.6%	95.5%	11	103.2%	96.9%
5	98.3%	97.0%	8	95.0%	94.7%	12	99.6%	93.7%
6	100.0%	100.0%	9	98.8%	95.8%	13	106.6%	99.1%
7	99.1%	98.8%	10	95.6%	93.2%	14	114.2%	100.0%
8	95.3%	94.7%	11	103.2%	97.0%	15	101.7%	100.1%
9	111.8%	100.0%	12	94.0%	95.1%	16	100.1%	103.9%

Comparisons: Efficiency of Design

Efficiencies of our designs relative to G-CEXCH and GA

One factor			Two factors			Three factors		
n	G-CEXCH	GA	n	G-CEXCH	GA	n	G-CEXCH	GA
3	100.0%	100.0%	6	97.5%	96.1%	10	102.0%	95.4%
4	97.4%	96.2%	7	97.6%	95.5%	11	103.2%	96.9%
5	98.3%	97.0%	8	95.0%	94.7%	12	99.6%	93.7%
6	100.0%	100.0%	9	98.8%	95.8%	13	106.6%	99.1%
7	99.1%	98.8%	10	95.6%	93.2%	14	114.2%	100.0%
8	95.3%	94.7%	11	103.2%	97.0%	15	101.7%	100.1%
9	111.8%	100.0%	12	94.0%	95.1%	16	100.1%	103.9%

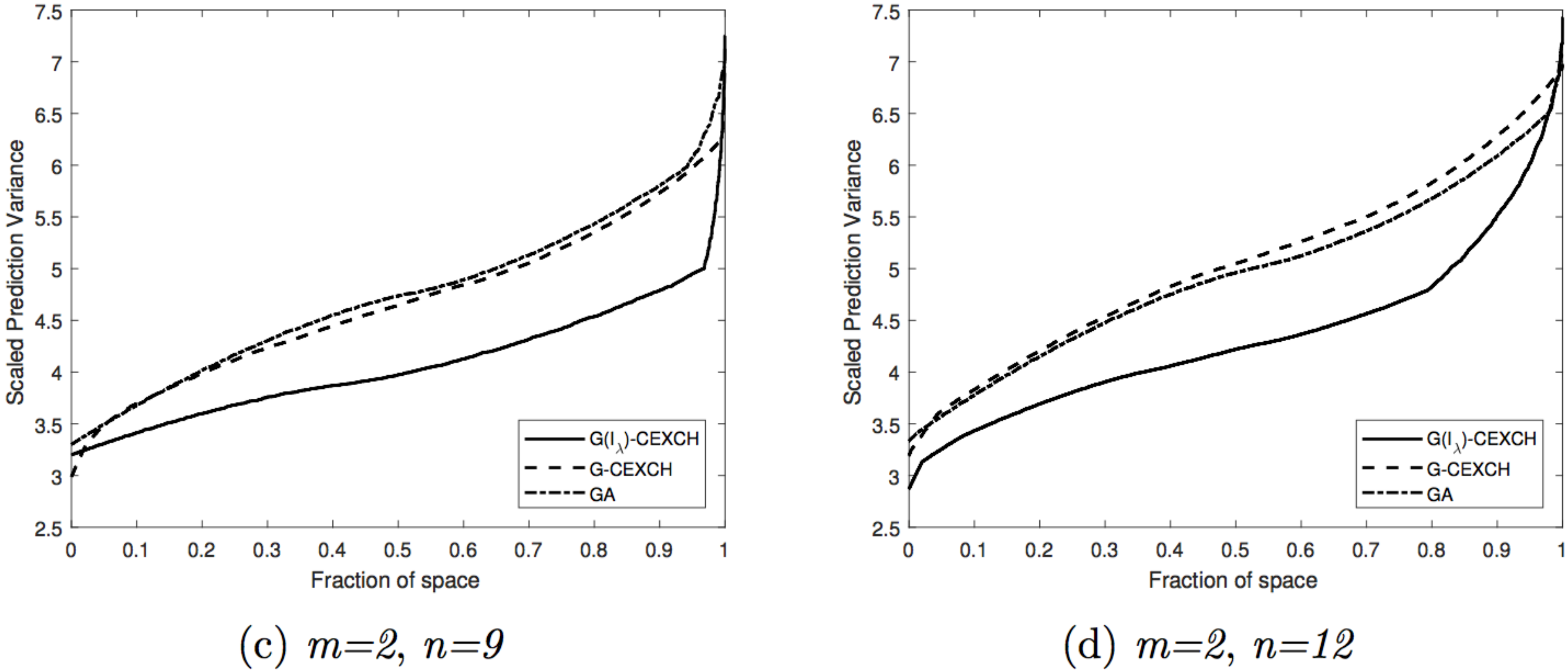
G-CEXCH and GA comparable

G-CEXCH and GA comparable

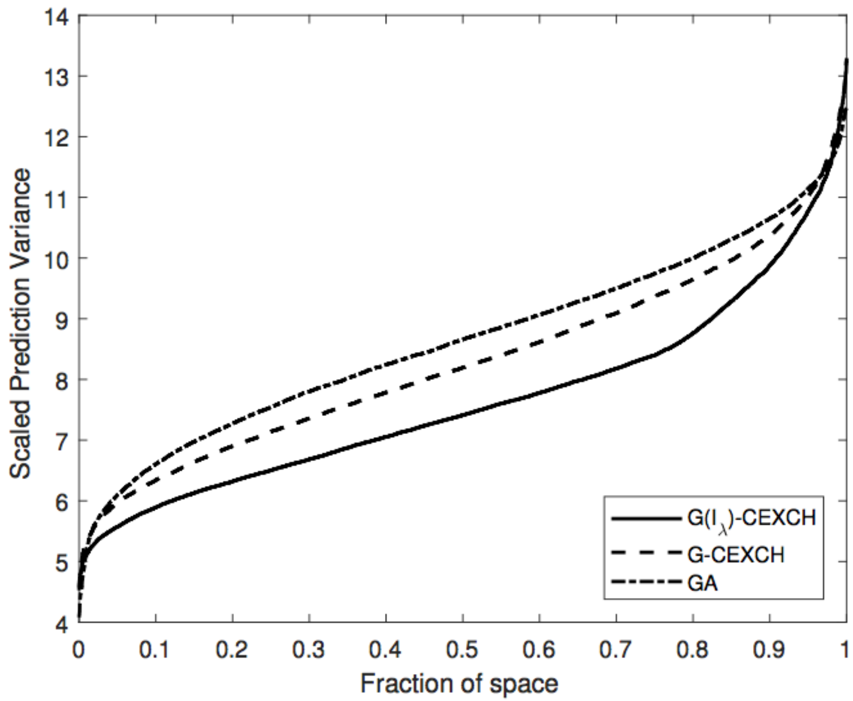
We're now beating G-CEXCH, sometimes GA

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

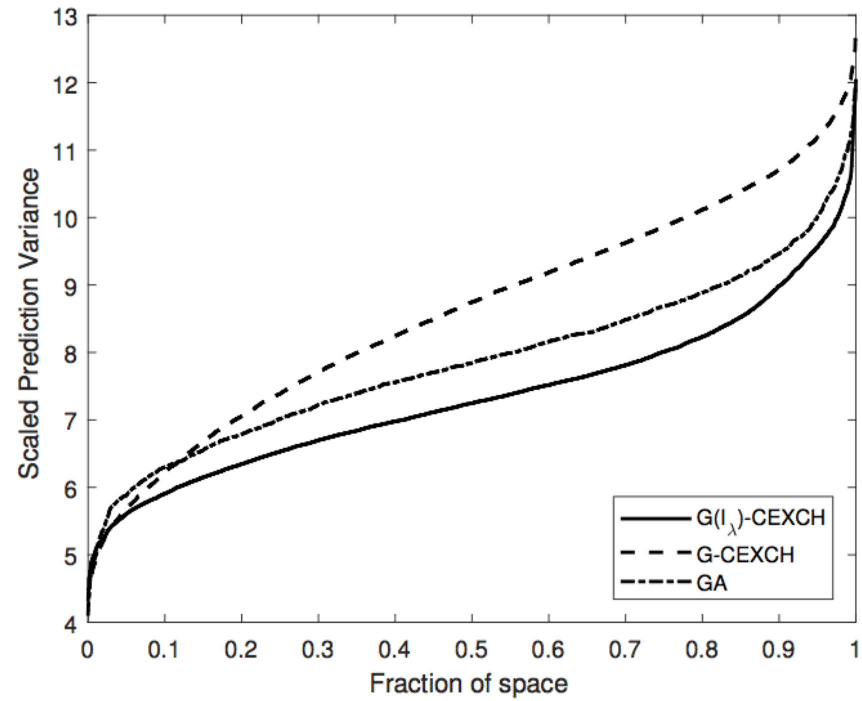
But these results are undersell our designs -- Consider the FDS plots – here 2D examples



FDS plots – 3D Examples



(e) m=3, n=12



(f) m=3, n=13

Now consider relative computing time

	One factor				Two factors				Three factors		
n	G(I_λ)-CEXCH	G-CEXCH	GA	n	G(I_λ)-CEXCH	G-CEXCH	GA	n	G(I_λ)-CEXCH	G-CEXCH	GA
3	1.00	0.12	6.18	6	1.00	1.15	16.18	10	1.00	6.33	47.26
4	1.00	0.06	2.92	7	1.00	0.72	10.75	11	1.00	6.27	42.85
5	1.00	0.05	2.40	8	1.00	1.43	18.96	12	1.00	6.69	45.21
6	1.00	0.14	5.18	9	1.00	3.19	39.30	13	1.00	4.97	34.53
7	1.00	0.07	2.36	10	1.00	1.42	18.56	14	1.00	8.96	56.16
8	1.00	0.05	2.28	11	1.00	1.54	20.09	15	1.00	8.14	51.02
9	1.00	0.06	2.50	12	1.00	0.77	9.79	16	1.00	9.30	57.07

One factor:

Best: G-CEXCH
Good: G-I_λ (us)
Bad!: GA

Two factors:

Best: G-I_λ (us)
Good: G-CEXCH
Bad!!: GA

Three factors:

Best: G-I_λ (us)
Good: G-CEXCH
BAD!!!: GA

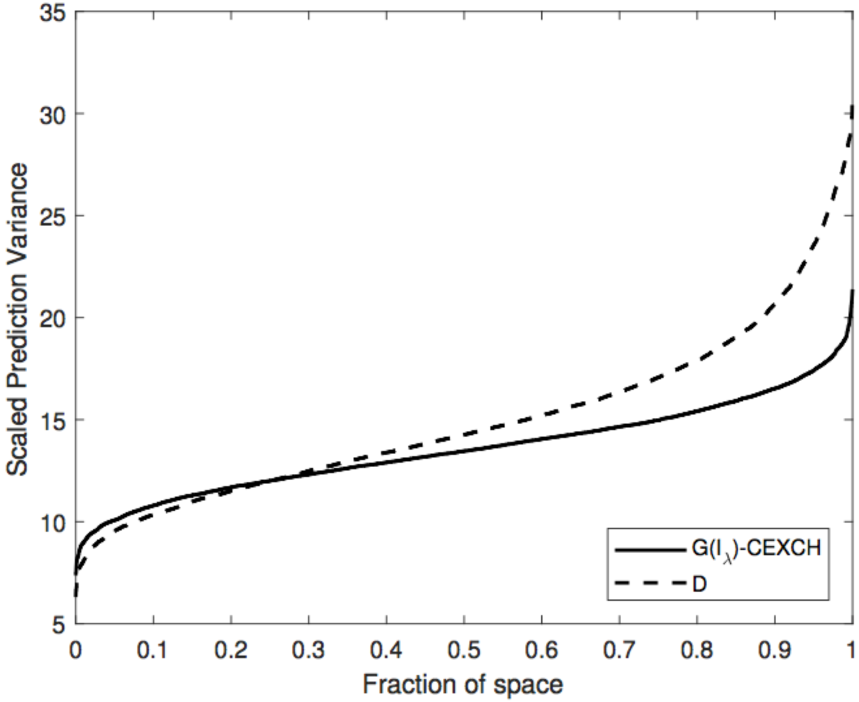
What about four and five factors?

- **G-CEXCH: Forget it**
  - **Predicted time for five factors: 166 days**
- **GA: Forget it**
  - **Predicted time for five factors: Hell freezes over**
- **Our algorithm: not a problem**

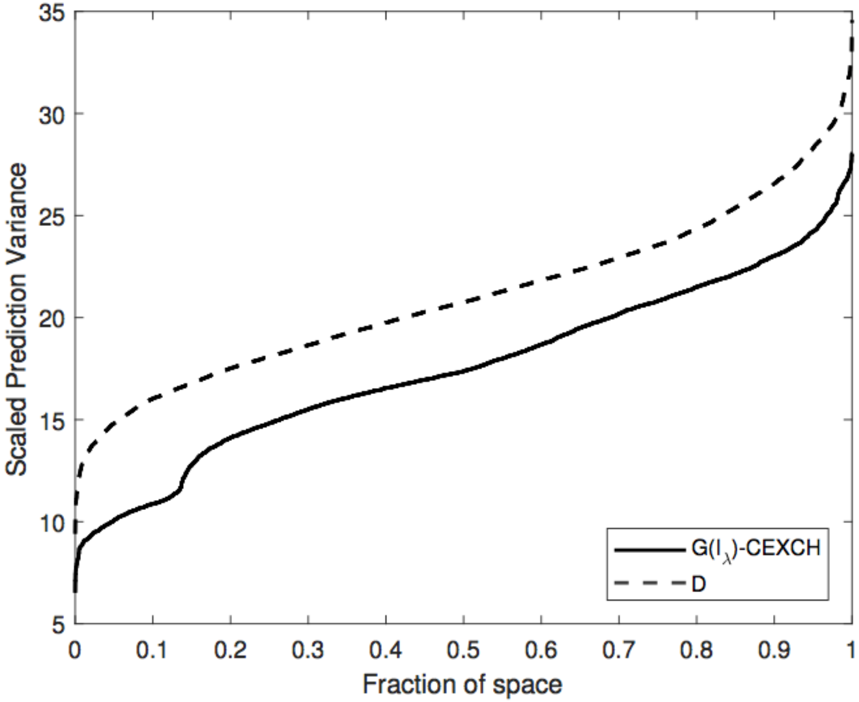
What about four and five factors?

- **G-CEXCH: Forget it**
  - **Predicted time for five factors: 166 days**
- **GA: Forget it**
  - **Predicted time for five factors: Hell freezes over**
- **Our algorithm: not a problem, but still 24 hours for 200 random starts---7 minutes per random start**

Our G-optimal designs vs D-optimal designs



(a) m=4, n=17



(b) m=5, n=23

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

Conclusions

- **Developed a (relatively) fast algorithm for computing G-optimal designs for quadratic models**
  - **Idea: Choose the W (moment) matrix for I-optimality such that the solution is G-optimal**
- **Much to be done:**
  - **Other models will present more complex W matrices**
  - **That means more w_i entries, more computing**
  - **Similarly for irregular design spaces**
  - **But--algorithm is linear in the number of w_i entries**

Thank you!

- **Compliments, congratulations:**
  - **nacht001@umn.edu**

- **Criticisms, complaints:**
  - **lucianhernandez@gmail.com**