

DEVELOPMENT AND APPLICATIONS OF
PARTICLE SWARM OPTIMIZATION
FOR CONSTRUCTING OPTIMAL EXPERIMENTAL DESIGNS

by

Stephen Joseph Walsh

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Statistics

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 30, 2021

©COPYRIGHT

by

Stephen Joseph Walsh

2020

All Rights Reserved

DEDICATION

To Meaghan.

ACKNOWLEDGEMENTS

First and foremost I express my humble gratitude to my advisor Dr. Borkowski. The mentorship, encouragement, guidance, problem space, expertise, and insights you have shared with me over the last few years have been absolutely invaluable and extremely appreciated. I look forward to carrying this torch of knowledge forward and sharing it with future generations.

Next, a very deep and heartfelt thank you to my committee members: Dr. Stacey Hancock, Dr. Albert Parker, and Dr. Dominique Zosso. It has been my honor and pleasure to know you and receive your teaching and advice throughout this journey. To know and work with such talented mathematical scientists is an absolute joy.

To Dr. Elizabeth Burroughs, Dr. Lisa Davis, Dr. Doreen Brown, the MTPEAKS program, Dr. Lillian Lin and SCRS, Dr. Roberta Amendola, Katie Sutich and numerous other professors, friends, and colleagues at MSU who are here unnamed: thank you all for your support, teaching, and for impacting my time as a Ph.D. student mid-career at Montana State University. This experience has fulfilled and surpassed all of my expectations.

Last, to Meaghan, Sarah, and Bridget: thank you for your support and friendship throughout the last four years. In large part because of you, I was able to finish this chapter in my professional life.

Thank you all,

Steve

Bozeman, MT, December 18, 2020

TABLE OF CONTENTS

1. A REVIEW OF PARTICLE SWARM OPTIMIZATION	1
1.1 Introduction.....	1
1.2 Basic PSO	3
1.3 Example Application: Optimizing the Rastrigin Function.....	7
1.4 History of PSO Development	8
1.5 PSO Algorithm and Standard Variants.....	15
1.5.1 Canonical PSO.....	15
1.5.2 SPSO 2007.....	15
1.5.3 SPSO 2011.....	16
1.6 Statistics: Optimization Problems and Possible Applica- tions of PSO	18
1.6.1 Parameter Estimation.....	19
1.6.1.1 Non-linear Models	20
1.6.1.2 Mixture Distributions	23
1.6.2 Optimal Design of Experiments	28
1.7 Conclusions.....	30
2. PRIMER ON OPTIMAL DESIGN OF EXPERIMENTS.....	41
2.1 Introduction.....	41
2.2 Classical vs. Optimal Designs	42
2.3 Useful Notation	45
2.4 Continuous vs. Exact Designs	47
2.4.1 Continuous Designs	47
2.4.2 Exact Designs	48
2.5 The General Equivalence Theorem	49
2.6 Some Optimality Criteria.....	51
2.6.1 D -optimality	51
2.6.2 G -optimality	52
2.6.3 A -optimality	54
2.6.4 IV -optimality	55
2.7 Brief Summary of Common Algorithms use to Generate Exact Optimal Designs	56
2.8 Conclusions.....	60
3. EXTENDING PARTICLE SWARM OPTIMIZATION TO GEN- ERATE EXACT OPTIMAL EXPERIMENTAL DESIGNS	62
ABSTRACT	62

TABLE OF CONTENTS – CONTINUED

3.1	Introduction.....	62
3.2	Extending the PSO Update Equations to Functions with Matrix Inputs	65
3.3	PSO-OptDoE: List of Julia Functions.....	67
3.3.1	swarmSearch_Hypercube	67
3.3.2	swarm_initialization	70
3.3.3	genRandDesign_fac	71
3.3.4	genLHC_fac	71
3.3.5	initialVelocity_2007	71
3.3.6	initialVelocity_2011	72
3.3.7	update_Vel	72
3.3.8	check_velocity	72
3.3.9	velLimit	72
3.3.10	check_confinement	73
3.3.11	confine_particle	73
3.3.12	update_pg_bests	73
3.3.13	update_ll_best	73
3.3.14	genNeighbors	74
3.3.15	randomSphere	74
3.3.16	D_criterion	74
3.3.17	A_criterion	75
3.3.18	genModelMat_fac	75
3.4	Software Validation and PSO Performance Benchmarking	75
3.4.1	Study Structure	76
3.4.2	Software Validation	77
3.4.2.1	<i>D</i> -optimal Designs Generated by PSO	78
3.4.2.2	<i>A</i> -optimal Designs Generated by PSO	82
3.4.2.3	Conclusions Regarding Software Validation	84
3.4.3	Further Performance Analysis of the PSO Algo- rithms for Generating Optimal Designs	85
3.5	Conclusions Regarding Feasibility of PSO to Generate Exact Optimal Designs	87
4.	FAST COMPUTATION OF HIGHLY EFFICIENT EXACT <i>G</i> - OPTIMAL DESIGNS VIA PARTICLE SWARM OPTIMIZATION AND PARALLEL COMPUTING	125
	ABSTRACT	125
4.1	Introduction.....	125
4.2	Scoring Candidate Designs on the <i>G</i> -optimality Scale.....	129

TABLE OF CONTENTS – CONTINUED

4.3	Small Exact PSO-Generated G -optimal Response Surface Designs	131
4.3.1	Results Summary	132
4.3.2	Comparison to State-of-the-Art Algorithms	132
4.3.3	New Small Exact G -optimal Response Surface Designs.....	134
4.4	Extending G -optimal design generating algorithms to $K=4,5$ factors	134
4.4.1	Comparison of $K=4,5$ factor G -PSO designs to published G -optimal designs	136
4.5	Conclusions.....	137
5.	A PARTICLE SWARM BASED ALGORITHM FOR GENER- ATING IV -OPTIMAL DESIGNS WITH SPECIFIED REPLICA- TION STRUCTURE	155
	ABSTRACT	155
5.1	Introduction.....	155
5.2	The IV -criterion	158
5.3	PSO-generated IV -optimal designs	160
5.3.1	Reproducing Borkowski (2003) Genetic Algorithm Generated Designs.....	160
5.3.2	$K = 4$ factor IV -optimal Designs	160
5.3.3	Comparison of the properties of the $K = 4$ PSO IV -optimal designs to the PSO G -optimal designs.	161
5.4	The importance of pure replication in designed experiments	164
5.4.1	Recommended estimate of σ^2 in designed experiments.....	166
5.4.2	The lack-of-fit (LOF) test.....	169
5.4.3	The Power of the LOF test.....	171
5.5	Using PSO to generate IV -optimal designs with a spec- ified replication structure.....	174
5.5.1	User specified replication structure in the PSO generating algorithm	175
5.5.2	An exhaustive study of some small K moderate N cases	177
5.5.2.1	$K = 2$, $N = 10$ IV -optimal designs with replication	177
5.5.2.2	$K = 2$, $N = 12$ IV -optimal designs with replication	179
5.5.2.3	$K = 3$, $N = 14$ IV -optimal designs with replication	180
5.5.2.4	$K = 3$, $N = 16$ IV -optimal designs with replication	181
5.5.2.5	$K = 4$, $N = 20$ IV -optimal designs with replication	181
5.5.3	Discussion.....	182
5.6	Conclusions and Future Work	185

TABLE OF CONTENTS – CONTINUED

6. A NEW PARTICLE SWARM OPTIMIZATION ALGORITHM BASED ON THE AITCHISON GEOMETRY FOR GENERATING EXACT OPTIMAL MIXTURE EXPERIMENT DESIGNS.....	237
ABSTRACT	237
6.1 Introduction.....	237
6.2 Adapting the Aitchison geometry to enable PSO on the Cartesian product of standard $(K-1)$ -simplices.....	243
6.2.1 Motivation.....	243
6.2.2 The Aitchison geometry.....	246
6.2.3 Inducing a vector space on Δ^{K-1}	247
6.2.3.1 Proof that $\oplus$ and $\otimes$ induce a vector space on Δ_O^{K-1} over $\mathbb{R}$	249
6.3 CprodSimplex-PSO: A new PSO algorithm to optimize matrix valued functions on the Cartesian product of standard K -simplices.....	256
6.3.1 CprodSimplex-PSO: velocity clamping	257
6.3.2 CprodSimplex-PSO: swarm initialization	258
6.3.3 CprodSimplex-PSO: particle confinement	259
6.3.4 CprodSimplex-PSO: PSO update equations	260
6.3.5 Discussion.....	261
6.4 Linear models for mixture experiments.....	261
6.4.1 First-order Scheffé model.....	262
6.4.2 Second-order Scheffé model.....	262
6.4.3 Special cubic model.....	262
6.4.4 Full cubic model.....	262
6.4.5 Special quartic model	262
6.4.6 Full quartic model.....	263
6.5 Optimality criteria for mixture designs	263
6.6 Validating CprodSimplex-PSO	264
6.6.1 (D -criterion) Generating simplex-lattice designs: CprodSimplex-PSO searches on the first and second- order Scheffé models.....	265
6.6.1.1 $K = 3$ first order model.....	266
6.6.1.2 $K = 3$ second order model.....	266
6.6.1.3 $K = 4$ first order model.....	266
6.6.1.4 $K = 4$ second order model.....	266
6.6.2 (D -criterion) Generating simplex-centroid designs: CprodSimplex-PSO searches on the special cubic model.....	267
6.6.2.1 $\{3, 3\}$ -SCD: special cubic model	268

TABLE OF CONTENTS – CONTINUED

6.6.2.2 $\{4, 3\}$ -SCD: special cubic model	268
6.6.3 (D -criterion) Augmented SCDs: CprodSimplex- PSO searches on the full cubic model	268
6.6.3.1 $K = 3$ full cubic model	269
6.6.3.2 $K = 4$ full cubic model	269
6.6.4 (IV -criterion) CprodSimplex-PSO searches for IV - optimal mixture designs.....	269
6.6.4.1 Second order model, $K = 3$, $N = 6$ runs.....	270
6.6.4.2 Second order model, $K = 3$, $N = 7$ runs.....	270
6.6.4.3 Second order model, $K = 3$, $N = 8$ runs.....	271
6.6.4.4 Second order model, $K = 3$, $N = 30$ runs.....	271
6.6.4.5 Second order model, $K = 4$, $N = 15$ runs.....	272
6.6.4.6 Special cubic model, $K = 4$, $N = 16$ runs.....	272
6.6.4.7 Special cubic model, $K = 4$, $N = 17$ runs.....	273
6.6.5 Discussion and conclusions.....	273
6.7 Experiments with lower and upper constraints on the mixture proportions.....	273
6.7.1 Functions to compute geometric summaries of $\diamond^{K-1}$	274
6.7.1.1 Checking and adjusting the consistency of $\diamond^{K-1}$	274
6.7.1.2 Calculating the number of D -dimensional sur- faces of $\diamond^{K-1}$	276
6.7.1.3 Calculating the extreme vertices and centroid of $\diamond^{K-1}$	277
6.7.1.4 Calculating the maximum distance between vertices.....	277
6.7.2 Additional subroutines required to support PSO searches on $\diamond^{K-1}$	278
6.7.2.1 Velocity clamping on $\diamond^{K-1}$	278
6.7.2.2 The truncated Dirichlet distribution: initial- izing a swarm of random design matrices on $\diamond^{K-1}$	278
6.7.2.3 Particle confinement on $\diamond^{K-1}$	283
6.8 Validating CprodSimplex-PSO on $\diamond^{K-1}$	284
6.8.1 Validation runs using the D -criterion	287
6.8.1.1 $K = 3$, $N = 9$, cubic model without 3-way interaction.....	287
6.8.1.2 $K = 3$, $N = 7$, second-order model	288
6.8.1.3 $K = 4$, $N = 8$, first-order model	288
6.8.1.4 $K = 4$, $N = 21$, first-order model	289
6.8.2 Validation runs using the IV -criterion.....	289
6.8.2.1 $N = 10$	290
6.8.2.2 Second order model, $K = 4$, $N = 15$ runs.....	291

TABLE OF CONTENTS – CONTINUED

6.9 Discussion and Conclusions.....	291
7. DISCUSSION, CONCLUSIONS, AND FUTURE WORK.....	323
7.1 Summary	323
7.2 Conclusions.....	324
7.3 Planned future work.....	325
REFERENCES CITED	328

LIST OF TABLES

Table	Page
1.1 Locations and objective function values of the local and global minimum of $h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$	23
3.1 Table indicating experimental levels of the criterion, PSVersion, and Swarm Size implemented in the validation study. The cell values indicate S , the levels of swarm size implemented in the validation study.....	89
3.2 Table indicating experimental levels of K (number of design factors), N (number of design points) and n_{run} (number of PSO runs implemented per design scenario.	90
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	108
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	109
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	110
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	111
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	112
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	113
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	114
3.3 Summary statistics from the validation study for PSO runs in generating D -optimal designs	115
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	115
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	116

LIST OF TABLES – CONTINUED

Table	Page
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	117
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	118
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	119
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	120
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	121
3.4 Summary statistics from the validation study for PSO runs in generating A -optimal designs	122
4.1 Relative efficiencies of G -optimal designs for each algorithm relative to GA generated designs.	140
4.2 Number of function evaluations across the $n_{\text{run}} = 140$ G -PSO runs using $S = 150$ particles.	141
4.3 New two-factor G -optimal designs generated by PSO. G -scores are shown for PSO designs and for corresponding GA generated designs.	142
4.4 New three-factor G -optimal designs generated by PSO for $N = 10, 11, 12, 13$. G -scores are shown for PSO designs and for corresponding GA generated designs.	143
4.5 New three-factor G -optimal designs generated by PSO for $N = 14, 15, 16$. G -scores are shown for PSO designs and for corresponding GA generated designs.	144

LIST OF TABLES – CONTINUED

Table	Page
4.6 G -efficiencies of published G -optimal designs for $K = 4, 5$ factors. The last column is the relative efficiency of the best G -PSO design compared to previously published designs. In all cases, PSO searches generated as or more efficient G -optimal designs (with some cases being much more efficient).	145
4.7 The G -PSO G -optimal design for $K = 4$, $N = 15$.	147
4.8 The G -PSO G -optimal design for $K = 4$, $N = 17$.	148
4.9 The G -PSO G -optimal design for $K = 4$, $N = 20$.	149
4.10 The G -PSO G -optimal design for $K = 4$, $N = 24$.	150
4.11 The G -PSO G -optimal design for $K = 5$, $N = 21$.	151
4.12 The G -PSO G -optimal design for $K = 5$, $N = 23$.	152
4.13 The G -PSO G -optimal design for $K = 5$, $N = 26$.	153
4.14 The G -PSO G -optimal design for $K = 5$, $N = 30$.	154
5.1 The $K = 4$, $N = 15$ IV -PSO generated design.	189
5.2 The $K = 4$, $N = 17$ IV -PSO generated design.	190
5.3 The $K = 4$, $N = 20$ IV -PSO generated design.	191
5.4 The $K = 4$, $N = 24$ IV -PSO generated design.	192
5.5 Comparison of the G and IV scores and relative efficiencies of the G -PSO and IV -PSO $K = 4$ factor designs.	193
5.6 t critical values for a size $\alpha = 0.05$ test as a function of df .	193
5.7 Full space of degree-of-freedom scenarios on ν_1 (LOF) and ν_2 (PE) for a design problem on $K = 3$ factors with $N = 16$ runs, intending to support the second order model which has $p = 10$ parameters.	201

LIST OF TABLES – CONTINUED

Table	Page
5.8 The $K = 2, 3, 4$ and N scenarios for which we implemented exhaustive study of generating IV -optimal designs with specified replication using PSO.	203
5.9 The $K = 4$ $N = 20$ IV -PSO generated design, replication scenario 1.	215
5.10 The $K = 4$ $N = 20$ IV -PSO generated design, scenario 11.	216
5.11 Replication scenarios and IV -scores of designs generated by PSO for $K = 2, N = 10$	217
5.12 Replication scenarios and IV -scores of designs generated by PSO for $K = 2, N = 12$	218
5.13 Replication scenarios and IV -scores of designs generated by PSO for $K = 3, N = 14$	219
5.14 Replication scenarios and IV -scores of designs generated by PSO for $K = 3, N = 16$	220
5.15 Replication scenarios and IV -scores of designs generated by PSO for $K = 4, N = 20$	221
6.1 The CprodSimplex-PSO generated IV -optimal design for $K = 3, N = 30$, second-order Scheffé model. N_i is the number of replicates of design point i	308
6.2 The CprodSimplex-PSO generated IV -optimal design for $K = 4, N = 15$, second-order Scheffé model. The design points from the coordinate exchange algorithm (CEXCH) of Goos et al. (2016) are in parenthesis. The CprodSimplex-PSO design points are not in parenthesis. Discrepancy between the two optimal designs are denoted with an asterisk.	309
6.3 The CprodSimplex-PSO generated IV -optimal design for $K = 4, N = 16$, special cubic model. The design matches that from the coordinate exchange algorithm (CEXCH) of Goos et al. (2016).	311

LIST OF TABLES – CONTINUED

Table	Page
6.4 The CprodSimplex-PSO generated <i>IV</i> -optimal design for $K = 4$, $N = 17$, special cubic model. The design points from the coordinate exchange algorithm (CEXCH) of Goos et al. (2016) are in parenthesis. The CprodSimplex-PSO design points are not in parenthesis. Discrepancy between the two optimal designs are denoted with an asterisk.....	312
6.5 The $K = 4$ $N = 21$, second-order model, with constraints <i>D</i> -PSO generated design.	318
6.6 The CprodSimplex-PSO generated <i>IV</i> -optimal design for $K = 4$, $N = 10$, second-order Scheffé model. <i>IV</i> -CH is the score reported by Coetzer and Haines (2017) and their design is shown in parentheses.....	319
6.7 The CprodSimplex-PSO generated <i>IV</i> -optimal design for $K = 4$, $N = 15$, second-order Scheffé model. <i>IV</i> -CH is the score reported by Coetzer and Haines (2017) and their design is shown in parentheses.....	321
6.8 Summary table of the performance of the new CprodSimplex-PSO algorithm to reproduce published <i>D</i> and <i>IV</i> -optimal designs for mixture experiments.....	322

LIST OF FIGURES

Figure	Page
1.1 Contour plot of the Rastrigin function for $n = 2$	32
1.2 Perspective plot of the Rastrigin function for $n = 2$	32
1.3 Application of bPSO to the Rastrigin function. The sequence of figures show the swarm state at the indicated iterations.	33
1.4 Example graphs illustrating different particle communication topologies for $S = 10$ particles. The labeling of the vertices represents the particle index.	34
1.5 Contour plot of SSE: $h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$	35
1.6 Four example sequences of a Newton-type optimization algorithm (represented by different colors). The black point indicates the starting value.	35
1.7 Scatterplot of the exponential decay model data and curves showing the fitted models for the global (blue) and local (red) minimum solutions.....	36
1.8 Application of SPSO 2007 to optimize $h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$. The algorithm converged to the global minimum at 54 iterations using only $S = 12$ particles.	37
1.9 Density curve for a sample of $n = 500$ points from a two-component Gaussian mixture with $\pi = 0.7$, $\mu_1 = 0$, $\mu_2 = 2.5$, and $\sigma_1 = \sigma_2 = 1$	38
1.10 Negative log-likelihood for a sample of $n = 500$ points from a two-component Gaussian mixture with $\pi = 0.7$, $\mu_1 = 0$, $\mu_2 = 2.5$, and $\sigma_1 = \sigma_2 = 1$	38
1.11 Three runs of the EM algorithm on the log-likelihood for the EM algorithm. The red point is the true parameter value. The blue EM chain reaches the global mode. The green chain terminates at a saddle point. Purple chain reaches a local mode.	39

LIST OF FIGURES – CONTINUED

Figure	Page
1.12 Application of SPSO 2011 to minimize the negative log-likelihood for the two-component Gaussian mixture. Particles were initialized in the range $[-10, 10]$ in both dimensions.....	40
3.1 $S = 50$ runs: Boxplots of PSO D -optimal designs for each scenario (top) and corresponding D -efficiencies (bottom).	91
3.2 $S = 150$ runs: Boxplots of PSO D -optimal designs for each scenario (top) and corresponding D -efficiencies (bottom).	92
3.3 $S = 500$ runs: Boxplots of PSO D -optimal designs for each scenario (top) and corresponding D -efficiencies (bottom).	93
3.4 Estimates of the probability that PSO successfully finds the D -optimal design for all PSO versions, swarm sizes, and design scenarios studied.	94
3.5 $S = 50$ runs: Boxplots of PSO A -optimal designs for each scenario (top) and corresponding A -efficiencies (bottom).	95
3.6 $S = 150$ runs: Boxplots of PSO A -optimal designs for each scenario (top) and corresponding A -efficiencies (bottom).	96
3.7 $S = 500$ runs: Boxplots of PSO A -optimal designs for each scenario (top) and corresponding A -efficiencies (bottom).	97
3.8 Estimates of the probability that PSO successfully finds the A -optimal design for all PSO versions, swarm sizes, and design scenarios studied.	98
3.9 D -optimal design: Median number of iterations to algorithm terminations as a function of K , N , S , and PSO algorithm.....	99

LIST OF FIGURES – CONTINUED

Figure	Page
3.10 <i>D</i> -optimal design: probability of success estimates as a function of PSO algorithm and swarm size.	100
3.11 <i>D</i> -optimal design: comparison of increase in probability of success of SPSO 2007 compared to Basic PSO at different swarm sizes.	101
3.12 <i>D</i> -optimal design: comparison of increase in probability of success of each Basic PSO and SPSO 2007 attributable to increase in swarm size ($S = 50$ is basis for comparison).	102
3.13 <i>A</i> -optimal design: Median number of iterations to algorithm terminations as a function of K , N , S , and PSO algorithm.	103
3.14 <i>A</i> -optimal design: probability of success estimates as a function of PSO algorithm and swarm size.	104
3.15 <i>A</i> -optimal design: comparison of increase in probability of success of SPSO 2007 compared to Basic PSO at different swarm sizes.	105
3.16 <i>A</i> -optimal design: comparison of increase in probability of success of each Basic PSO and SPSO 2007 attributable to increase in swarm size ($S = 50$ is basis for comparison).	106
3.17 <i>D</i> -optimal design: Boxplots of number of iterations to termination of each PSO algorithm across the study space. The dotted blue line is at 1000 iterations while the solid blue line is at 5000 iterations.	123
3.18 <i>A</i> -optimal design: Boxplots of number of iterations to termination of each PSO algorithm across the study space. The dotted blue line is at 1000 iterations while the solid blue line is at 5000 iterations.	124

LIST OF FIGURES – CONTINUED

Figure	Page
4.1 SPV surfaces of the second-order model for the G , D , IV , and random designs for the $K = 2$, $N = 11$ design scenario. The scaling of the z -axis is the same for sub-figures (a), (b), and (c) and so colors can be directly compared in these figures. The G -score of a design is the maximum SPV over $\mathcal{X}$ (i.e., the plot square).	138
4.2 The top set of panels shows boxplots of the $n_{\text{run}} = 140$ G -efficiencies of the designs generated by PSO. The red dot represents the G -eff of the GA generated design from Borkowski (2003). The bottom set of panels show boxplots of the G -relative efficiency of the PSO generated designs vs. the GA designs.	139
4.3 The top set of panels shows boxplots of the $n_{\text{run}} = 210$ G -efficiencies of the designs generated by PSO. The red dot represents the G -eff of the coordinate exchange algorithms generated design from Rodrigues et. al. (2010) and Hernandez and Nachtsheim (2018). The bottom set of panels show boxplots of the G -releff of the PSO generated designs with the G -score of the coordinate exchange generated designs used in the denominator of the relative efficiency calculation.	146
5.1 The top set of panels shows boxplots of the $n_{\text{run}} = 140$ IV -scores of the designs generated by PSO presented on the scale of Borkowski (2003) (and so larger IV -score on this scale represents a better design). The red dot represents the IV -score of the GA generated design from Borkowski (2003). The bottom set of panels show boxplots of the IV -efficiencies of the PSO generated designs with the IV -score of the GA design used in the denominator of the efficiency calculation.	187

LIST OF FIGURES – CONTINUED

Figure	Page
5.2 The top panel shows boxplots of the $n_{\text{run}} = 250$ IV -scores of the $K = 4$ designs generated by PSO using the unscaled IV -criterion (and so smaller score indicates a better design). The middle panel shows the IV -efficiency computed across all N . The bottom panel shows the IV -efficiency computed conditioned on N	188
5.3 FDS plots for each $K = 4$, N comparing the G -PSO to IV -PSO design. The horizontal dotted lines show the IV -scores of each design.....	194
5.4 CSPV plots for $K = 4$, $N = 15$ comparing the G -PSO to IV -PSO design.....	195
5.5 CSPV plots for $K = 4$, $N = 17$ comparing the G -PSO to IV -PSO design.....	196
5.6 CSPV plots for $K = 4$, $N = 20$ comparing the G -PSO to IV -PSO design.....	197
5.7 CSPV plots for $K = 4$, $N = 24$ comparing the G -PSO to IV -PSO design.....	198
5.8 Contour plots of the face of $\mathcal{X}$ defined by $x_3 = -1$, $x_4 = 1$ for $N = 24$ (a) the four factor G -PSO design and (b) the four factor IV -PSO design.	199
5.9 CFDS plot of the G and IV -PSO designs for face ($x_3 = -1, x_4 = 1$). The dashed lines are the global IV -scores for both designs.....	200
5.10 CCD with two center replicates for $K = 3$ factors and $N = 16$ design points.....	202
5.11 Power curves for the LOF test under the df scenarios shown in Table 5.7.....	203
5.12 $K = 2$, $N = 10$ study: IV -PSO designs with specified replication corresponding to scenarios 1, 3, 4, and 6.....	204

LIST OF FIGURES – CONTINUED

Figure	Page
5.13 $K = 2$, $N = 10$ study: FDS plots of IV -PSO designs with specified replication corresponding to scenarios 1, 3, 4, and 6.	205
5.14 SPV surfaces for the $K = 2$ $N = 10$ selected IV -PSO design replication scenarios.	206
5.15 $K = 2$, $N = 12$ study: IV -PSO designs with specified replication corresponding to scenarios 1, 6, 10 and 14.1.....	207
5.16 $K = 2$, $N = 12$ study: FDS plots of IV -PSO designs with specified replication corresponding to scenarios 1, 6, 10 and 14.1.	208
5.17 SPV surfaces for the $K = 2$ $N = 12$ selected IV -PSO design replication scenarios.	209
5.18 $K = 3$, $N = 14$ study: IV -PSO designs with specified replication corresponding to scenarios 1, 4, 5 and 7.	210
5.19 $K = 3$, $N = 14$ study: FDS plots of IV -PSO designs with specified replication corresponding to scenarios 1, 4, 5 and 7.	211
5.20 $K = 3$, $N = 16$ study: IV -PSO designs with specified replication corresponding to scenarios 1, 5, 11, and 15.	212
5.21 $K = 3$, $N = 16$ study: FDS plots of IV -PSO designs with specified replication corresponding to scenarios 1, 5, 11, and 15.....	213
5.22 $K = 4$, $N = 20$ study: FDS plots of IV -PSO designs with specified replication corresponding to scenarios 1 and 11.	214
6.1 Simplex lattice designs for $K = 3$ and $m = 1, 2, 3, 5, 7, 9$	294
6.2 Results of the CprodSimplex-PSO search: $K = 3$, $N = 3$, first-order model.	295
6.3 Results of the CprodSimplex-PSO search: $K = 3$, $N = 6$, second-order model.	296

LIST OF FIGURES – CONTINUED

Figure	Page
6.4 Results of the CprodSimplex-PSO search: $K = 4$, $N = 4$, second-order model.	297
6.5 Results of the CprodSimplex-PSO search: $K = 4$, $N = 10$, second-order model.	298
6.6 Simplex-centroid design for $K = 3$ factors.	299
6.7 Results of the CprodSimplex-PSO search: $K = 3$, $N = 7$, special cubic model.	300
6.8 Results of the CprodSimplex-PSO search: $K = 4$, $N = 14$, special cubic model.	301
6.9 Results of the CprodSimplex-PSO search: $K = 3$, $N = 10$, full cubic model.	302
6.10 Results of the CprodSimplex-PSO search: $K = 4$, $N = 20$, full cubic model.	303
6.11 Results of the CprodSimplex-PSO search: $K = 3$, $N = 6$, second order model.	304
6.12 Results of the CprodSimplex-PSO search: $K = 3$, $N = 7$, second order model.	305
6.13 Results of the CprodSimplex-PSO search: $K = 3$, $N = 8$, second order model.	306
6.14 Results of the CprodSimplex-PSO search: $K = 3$, $N = 30$, second order model.	307
6.15 Results of the CprodSimplex-PSO search: $K = 4$, $N = 15$, second order model.	308
6.16 Results of the CprodSimplex-PSO search: $K = 4$, $N = 16$, special cubic model.	310
6.17 Results of the CprodSimplex-PSO search: $K = 4$, $N = 17$, special cubic model.	310

LIST OF FIGURES – CONTINUED

Figure	Page
6.18 Left panel shows a sample using uniform Dirichlet parameters and the probability integral transform approach. The right hand panel shows a sample using uniform Dirichlet parameters and the Gibbs sampling approach.	313
6.19 Results of the CprodSimplex-PSO search: $K = 3$, $N = 9$, modified cubic model (no 3-way interaction) and constraint $x_1 \leq 0.5$	314
6.20 Results of the CprodSimplex-PSO search: $K = 3$, $N = 7$, second-order model and constraints.	315
6.21 Results of the CprodSimplex-PSO search: $K = 4$, $N = 8$, first-order model and constraints.	316
6.22 Results of the CprodSimplex-PSO search: $K = 4$, $N = 21$, second-order model and constraints.	317
6.23 Results of the CprodSimplex-PSO search: $K = 4$, $N = 10$, quadratic Scheffé polynomial and constraints.	317
6.24 Results of the CprodSimplex-PSO search: $K = 4$, $N = 15$, quadratic Scheffé polynomial and constraints.	320

LIST OF ALGORITHMS

Algorithm	Page
1.1 The Basic PSO Algorithm.....	5
1.2 confine: Absorbing wall particle confinement.....	7
1.3 confine: Reflecting wall particle confinement.	18
6.4 CprodSimplex-PSO: random uniform swarm initial- ization of positions.....	259
6.5 CprodSimplex-PSO: PSO update equations for par- ticle i	260
6.6 CprodSimplex-PSO on $\diamond^{K-1}$: random uniform swarm initialization of positions.	284
6.7 CprodSimplex-PSO on $\diamond^{K-1}$: confine particles to the boundary (edges and faces) defined by $\diamond^{K-1}$	285

NOMENCLATURE

ijk	most common symbols for indices
N	Number of design points
M	Number of unique design points
K	Number of experimental factors
S	Number of particles in a swarm
$\mathbb{R}$	the real numbers
$\mathbb{N}$	the natural numbers
$\times_{j=1}^K \mathbb{R} = \mathbb{R}^K$	K -dimensional Cartesian product of real numbers
$\mathcal{X} = [-1, 1]^K$	K -dimensional hypercube; design space, PSO search space
$\mathbf{x}' \in \mathcal{X}$	$1 \times K$ dimensional row-vector; design point
$\mathbf{X} \in \mathcal{X}^N$	$N \times K$ exact design matrix
$\mathbf{X}'$	transpose of matrix $\mathbf{X}$
Ξ	space of continuous designs
ξ	a specific continuous design (i.e.as a probability measure)
p	Number of parameters in a model
$\boldsymbol{\beta} : p \times 1$	parameter vector for a linear model
$\mathbf{f}'(\mathbf{x}')$	$1 \times p$ model vector for design point $\mathbf{x}'$
$\mathbf{F}(\mathbf{X}), \mathbf{F}$	$N \times K$ model matrix (contains columns of $\mathbf{X}$)
$\mathbf{H}$	the hat matrix in the ordinary least-squares estimation setting
f	generic objective function for optimization problems
$\mathbf{I}_K$	identity matrix of dimension K
$\mathbf{1}_K$	$K \times 1$ vector of 1s
$\mathbf{J}_K$	$K \times N$ matrix of 1s
$\mathbf{e}_i$	vector of 0s with a 1 in the i th position
$D(\mathbf{X})$	D -criterion score for design $\mathbf{X}$
$IV(\mathbf{X})$	Integrated-variance criterion score for design $\mathbf{X}$
$G(\mathbf{X})$	G -criterion score for design $\mathbf{X}$
$A(\mathbf{X})$	A -criterion score for design $\mathbf{X}$
$\boldsymbol{\theta}$	general symbol for a parameter of a statistical model
Ω	general symbol for parameter space
$E(X)$	expected value of random variable X (will have dimension of X)
$\text{Var}(X)$	variance of random variable X (scalar or matrix)

NOMENCLATURE – CONTINUED

SSE	sum-of-squared residual errors
PE	pure error
$SSPE$	sum-of-squared pure error
LOF	lack-of-fit
$SSLOF$	sum-of-squares attributable to LOF
$\mathcal{N}(\mu, \sigma^2)$	normal (Gaussian) distribution, mean μ , variance σ^2
$\mathcal{N}_N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate Gaussian distribution of dimension N
$\boldsymbol{\mu}$	$N \times 1$ mean vector (parameter)
$\boldsymbol{\Sigma}$	$N \times N$ variance-covariance matrix (parameter)
$U(0, 1)$	uniform 0,1 distribution
$U_K(\mathbf{0}, \mathbf{1})$	multivariate uniform distribution of dimension K , all bounds are (0,1)
$\mathbf{U} = \{u_{nk}\}$	$N \times K$ uniform random matrix with $u_{nk} \stackrel{\text{iid}}{\sim} U(0, 1)$
$\text{Dir}(\boldsymbol{\theta} \boldsymbol{\alpha})$	Dirichlet distribution on random variable $\boldsymbol{\theta}$ with parameter $\boldsymbol{\alpha}$
$\prod \text{Dir}(\boldsymbol{\theta} \boldsymbol{\alpha})$	product Dirichlet distribution
H_0	null hypothesis
H_A	alternative hypothesis
α	the significance level for a hypothesis test
ν	degrees-of-freedom, also DF
$t_\nu^{1-\frac{\alpha}{2}}$	$1 - \frac{\alpha}{2} \times 100$ quantile of Student's t -distribution with ν DF
$F_{\nu_1, \nu_2}^{1-\alpha}$	$(1 - \alpha) \times 100$ quantile of Snedecor's F -distribution with ν_1, ν_2 DF
Δ^{K-1}	The standard $(K - 1)$ -simplex, closed set; also the design space for mixture experiments
Δ_O^{K-1}	The standard $(K - 1)$ -simplex, open set
$\times_{j=1}^N \Delta^{K-1}$	space of design matrices for optimal mixture design searches
$\diamond^{K-1} \subset \Delta^{K-1}$	convex irregular hyperpolytope
$\oplus$	matrix direct sum
$\odot$	Hadamard product (or exponent).
$\oplus$	vector addition under the Aitchison geometry
$\otimes$	multiplication of a scalar by a vector under the Aitchison geometry
SLD	simplex lattice design
$\{K, m\}$	SLD of K components with m grid points
SCD	simplex centroid design

ABSTRACT

The primary objective motivating this dissertation was to illustrate the efficacy of particle swarm optimization (PSO) as the engine of an algorithm to generate optimal design of experiments (DoE). PSO is a wildly popular and successful meta-heuristic and machine learning algorithm which makes no assumptions regarding the behavior of the function being optimized. Optimal DoE, in part thanks to modern computing, has become the current dominant paradigm for practitioners to generate a DoE with some desirable property. We bring together these concepts first by extending the PSO to optimizing functions that take matrix inputs. `Julia` software was developed for this purpose and validated against published results. A detailed benchmarking study of three PSO variants was conducted and a preferred version of the algorithm was identified for further research and application. Next, we implemented the approach to generating G -optimal designs—a difficult minimax optimization problem. New heretofore unknown G -optimal designs have been produced and the efficacy of PSO in generating efficiently (w.r.t. computing time) highly G -optimal designs is compared to current published results. Next, a new algorithm for generating optimal designs with specified replication structure, and thereby guaranteed a degrees-of-freedom for estimating the pure error variance, is proposed, illustrated, benchmarked and validated. Last, we propose a new algorithm for generating optimal mixture experiment designs which implements a PSO type search using a non-Euclidean geometry (specifically the Aitchison geometry). In this setting the space of candidate matrices is the Cartesian product of standard $(K - 1)$ -simplices. The algorithm is extended to mixture experiments with lower and upper constraints on the mixture proportions; in this setting, the space of candidate matrices is the Cartesian product of high-dimensional irregular convex polytopes. The algorithm is validated against very recent published results. In total, the work presented in this dissertation speaks very favorably to PSO as a tool for generating optimal DoEs. We believe this approach should become part of the standard machine learning and statistical tool box for generating optimal experimental designs.

CHAPTER ONE

A REVIEW OF PARTICLE SWARM OPTIMIZATION

1.1 Introduction

Particle Swarm Optimization (PSO) belongs to the class of metaheuristic optimization algorithms which include (among others): genetic algorithms (GA), simulated annealing, differential evolution, ant colony optimization, harmony search, and cuckoo search. As a metaheuristic it makes few assumptions about structure or properties of the objective function f . Ownership of the algorithm as a mathematical tool is typically claimed by the computational science, artificial intelligence, and/or machine learning fields. We note that the progenitors of the algorithm, however, James Kennedy and Russel C. Eberhart, are a social psychologist and an electrical engineer, respectively [57].

PSO is an agent-based modeling approach to optimization. Agent-based approaches are modeling regimes which aim to simulate social behavior by parameterizing individual agents with respect to how they interact with each other and the environment. The inspiration behind the development of PSO was to produce a model that could mimic via simulation the flocking behavior of animals (e.g., flocks of birds or schools of fish). During initial simulations, the progenitors observed that the particles behaved more like a swarm than a flock and were apparently performing optimization. This observation spawned a tremendous amount of algorithm development and application in the last 24 years. As of September 12, 2019 at 11:44PM MST, a Google search for the seminal “Particle Swarm Optimization”

paper by Kennedy and Eberhart indicates that the publication currently has 55191 citations.

The attractiveness of PSO as a candidate algorithm for solving optimization problems (in this author's opinion) is as follows:

1. Elegance. PSO is simple. The intelligence of the algorithm relies primarily on two operations: vector scaling and vector addition.
2. Computing is cheap. Publications on algorithm development and published computer codes for implementing the algorithm have been extremely numerous since the 2000s. Multiple scientific programming languages have a library to perform PSO for functions that evaluate vector inputs, e.g., Matlab, Python, and R.
3. PSO is robust to local optima entrapment. Therefore, it is a good approach when the objective function is non-convex and multimodal.
4. PSO makes no assumptions about the function being optimized. PSO does not use the gradient and so it does not require that the function be continuous or differentiable.
5. PSO has been demonstrated to perform well up to dimension 50 with a relatively small number of particles (about 40). The curse of dimensionality is a problem for all who wish to optimize functions in high dimensional spaces, and this is a setting where PSO has been observed to perform "bizarrely well".

There has been limited application of PSO in solving statistics problems. Most of these applications have been to problems in optimal design of experiments. The applications, while important, have focused on theoretically optimal continuous

designs, and little attention has been given to exact designs. For this reason, we are interested in pursuing opportunities to raise awareness among the statistics community regarding PSO, its benefits and shortcomings, and potential applications. PSO may indeed prove highly valuable to the statistics community as a go-to method for optimization, especially in high-dimensional and mathematically intractable problems.

In the following sections, we formally describe the mainstay developments of the PSO algorithm in the last 24 years. We focus on the developments of what the core computational science community refers to as *standard particle swarm optimization* (SPSO). Indeed, as aforementioned, the seminal paper alone has over 55,000 citations—we cannot possibly review all developments in the field. Thus, we will keep our focus to the SPSO vein of algorithms as these are (at least were intended to be) benchmark algorithms against which research and improvements should be gauged.

1.2 Basic PSO

We present the basic PSO (bPSO) pseudocode in Algorithm 1.1. The notation used in this section is as follows. The objective is to compute

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} f(\mathbf{x}),$$

where f is a scalar valued function and the feasible region may be an irregular hypercube $\mathcal{X} = \times_{k=1}^K [l_k, u_k] \subset \mathbb{R}^K$. We group the lower bounds and upper bounds into vectors $\mathbf{l}_b = \{l_k\}$ and $\mathbf{u}_b = \{u_k\}$. Particle i 's position is denoted $\mathbf{x}_i \in \mathcal{X}$ and its velocity is denoted $\mathbf{v}_i$. The notation $\mathbf{x} \sim U_K(\mathbf{l}_b, \mathbf{u}_b)$ means $\mathbf{x}$ is drawn from a multivariate uniform distribution of dimension K with respective lower and upper

bounds $\mathbf{l}_b$ and $\mathbf{u}_b$, i.e., this is how particles are initially placed in $\mathcal{X}$. The symbol S denotes the number of particles and so $i = 1, 2, \dots, S$. For benchmarking bPSO, the number of particles is typically set at $S = 40$.

The primary intelligence of PSO lies in the velocity and position update equations (lines 22 and 24 of Algorithm 1.1) which we restate here as a function of iteration index t :

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) \quad (\text{inertia}) \quad (1.1)$$

$$+ c_1 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{p}_{\text{best},i} - \mathbf{x}_i(t)) \quad (\text{cognitive}) \quad (1.2)$$

$$+ c_2 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{g}_{\text{best}} - \mathbf{x}_i(t)) \quad (\text{social}) \quad (1.3)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (1.4)$$

where $\odot$ represents the Hadamard product (elementwise multiplication).

The velocity update equation consists of three components. The first term, shown in (1.1), is the particle’s current velocity scalar term ω , and is often referred to as the particle’s ‘inertia’ as this component governs how resistant the particle is to changing its current velocity. The second term (1.2) is called the cognitive component and it governs the portion of the velocity update that will be influenced by the particles ‘memory’ of the best position in $\mathcal{X}$ it has visited up to time t —this ‘personal best’ position is denoted $\mathbf{p}_{\text{best},i}$. The cognitive component is weighted by tuning parameter c_1 . The third term (1.3) represents communication between the particles in the swarm represented by the sharing of the swarm’s ‘global best’ known position $\mathbf{g}_{\text{best}}$ —this term is often referred to as the ‘social’ component of the velocity update. Similar to the cognitive component, the social component is scaled by tuning constant c_2 . Both the cognitive and social component are, element-wise, perturbed by a random uniform number in $[0, 1]$. This introduces stochastic behavior into the

Algorithm 1.1: The Basic PSO Algorithm

```

1: Input: Objective function  $f$ , number of particles  $S$ , search space bounds  $\mathbf{l}_b$  and  $\mathbf{u}_b$ 
2: Output: Particle swarm solution (global best position)  $\mathbf{g}_{\text{best}}$ 
3: // Initialize particle  $\mathbf{x}_i$ 's position, velocity, and fitness
4: for each  $i = 1, \dots, S$  do
5:    $\mathbf{x}_i \sim U_K(\mathbf{l}_b, \mathbf{u}_b)$ 
6:    $\mathbf{v}_i \sim U_K\left(\frac{\mathbf{l}_b - \mathbf{x}_i}{2}, \frac{\mathbf{u}_b - \mathbf{x}_i}{2}\right)$ 
7:    $\{v_{ik} \leftarrow \min\{v_{ik}, v_k^{\max}\}\}$ 
8:    $\mathbf{p}_{\text{best},i} \leftarrow \mathbf{x}_i$ 
9: endfor
10:  $\mathbf{g}_{\text{best}} \leftarrow \underset{\mathbf{x}_i \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_S\}}{\text{argmin}} f(\mathbf{x}_i)$ 
11: // Swarm search loops
12: while stopping criteria not met do
13:   for each  $i = 1, \dots, S$  do
14:     // Update velocities
15:      $\mathbf{v}_i \leftarrow \omega \mathbf{v}_i + c_1 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{p}_{\text{best},i} - \mathbf{x}_i) + c_2 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{g}_{\text{best}} - \mathbf{x}_i)$ 
16:      $\{v_{ik} \leftarrow \min\{v_{ik}, v_k^{\max}\}\}$ 
17:   endfor
18:   for each  $i = 1, \dots, S$  do
19:     // Update positions and fitness
20:      $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$ 
21:      $\mathbf{x}_i \leftarrow \text{confine}(\mathbf{x}_i)$ 
22:     // Update fitness and knowledge of  $f$  given new positions
23:     if  $f(\mathbf{x}_i) < f(\mathbf{p}_{\text{best},i})$ 
24:        $\mathbf{p}_{\text{best},i} \leftarrow \mathbf{x}_i$ 
25:       if  $f(\mathbf{p}_{\text{best},i}) < f(\mathbf{g}_{\text{best}})$ 
26:          $\mathbf{g}_{\text{best}} \leftarrow \mathbf{p}_{\text{best},i}$ 
27:       endif
28:     endif
29:   endfor
30: endwhile
31: return

```


search. The best values of tuning parameters ω , c_1 , and c_2 have been the subject of numerous studies which we will briefly discuss in subsequent sections. For bPSO these constants take values

$$\omega = 0.72984, \quad \text{and} \quad c_1 = c_2 = 2.05\omega = 1.496172.$$

The position update shown in Eq. (1.4) is simply displacement of particle i 's position by adding the updated velocity.

Additional aspects of the algorithm are worth some discussion. Lines 7 and 16 in Algorithm 1.1 show that velocities in each dimension are limited to a parameter $v_k^{\max}$. This is typically taken as the length of the search space in dimension k . The purpose of this limiting is to ensure particles do not often ‘warp’ completely across $\mathcal{X}$ in an iteration, i.e., $v_k^{\max}$ is a stepsize parameter. The **if** statements in lines 23-28 in Algorithm 1.1 represent the knowledge updating of the algorithm and thus how the particle's best position $\mathbf{p}_{\text{best},i}$ and the swarm's best position $\mathbf{g}_{\text{best}}$ are checked for each particle i at each iteration and updated if an improvement to fitness of the objective is found.

Line 18 in Algorithm 1.1 is the confinement subroutine. The purpose of this function is to check particle positions and adjust any particles that flew outside of $\mathcal{X}$ at this iteration of the algorithm. The basic PSO algorithm uses an ‘absorbing wall’ confinement as shown in Algorithm 1.2. This confinement is referred to as ‘absorbing wall’ because, in the event that the particle leaves $\mathcal{X}$, it is mapped to the boundary (in the respective dimensions that it is not within the search space) and the particle's velocity is set to 0 in that dimension (thus its velocity is absorbed by the wall).

Last, the main swarm search (i.e., the **while** loop in Algorithm 1.1) continues until some user-specified stopping criteria is satisfied. Many stopping criteria have

Algorithm 1.2: confine: Absorbing wall particle confinement.

- 1: **Input:** Particle i 's position $\mathbf{x}_i$
- 2: **Output:** A valid particle position in $\mathcal{X}$ and an adjustment to the particle's velocity $\mathbf{v}_i$
- 3: // Check each dimension independently
- 4: **if** $\{x_{ik} \notin (l_k, u_k)\}$
- 5: $\{v_{ik} \leftarrow 0\}$
- 6: **if** $\{x_{ik} < l_k\}; \{x_{ik} \leftarrow l_k\}$ **endif**
- 7: **if** $\{x_{ik} > u_k\}; \{x_{ik} \leftarrow u_k\}$ **endif**
- 8: **endif**
- 9: **return**

been implemented in the literature. Common choices include: maximum number of iterations, convergence of all particles in the swarm to a single point or to a radius of specified length, and convergence of the best known solution $\mathbf{g}_{\text{best}}$ (sometimes taken as a non-zero relative tolerance because the swarm knowledge regarding $\mathbf{g}_{\text{best}}$ can stagnate for a large number of iterations).

1.3 Example Application: Optimizing the Rastrigin Function

The Rastrigin function is a multivariate function of cosines. It is a typical test case function for mathematical optimization because it has a multitude of local optima but only a single global optima. The function definition is

$$f(\mathbf{x}) = AK + \sum_{k=1}^K (x_k^2 - A \cos(2\pi x_k)), \quad (1.5)$$

where $A = 10$, and the search space is a finite K -dimensional hypercube $\mathcal{X} = [-5.12, 5.12]^K$. The global minimum is $\min f = 0$, and the minimizer is $\mathbf{0} = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} f(\mathbf{x})$. We include a contour plot and perspective plots of the Rastrigin function

in Figures 1.1 and 1.2, respectively.

We prototyped the bPSO algorithm in the R statistical programming language [76]. We set the stopping criterion as a non-zero change in $\mathbf{g}_{\text{best}}$ with relative tolerance equal to the square root of machine epsilon (value of $1.490\,116 \times 10^{-8}$). The algorithm achieved the stopping criterion in 114 iterations and the swarm found the global minimum. We include a set of graph panels of the swarm at various iterations in Figure 1.3 to illustrate the behavior of the swarm. In this graphic, the white points represent the particles. The arrows extending from the points represent the displacement of the particle at this iteration (i.e., the quantity $\mathbf{x}_i + \mathbf{v}_i$), so at the next iteration the particle jumps to the head of the arrow. Sub-figure (d) in this set of graphics indicates that by iteration 69 the particle search radius is already quite small and focused on the global minimizer at $(0,0)'$. The last panel (f) at iteration 114 (the last iteration) indicates that the convergence criteria was met, but all particles did not find the global minimum. Having all particles finding the global optima is not a necessary condition for a successful solution.

1.4 History of PSO Development

In this section we briefly discuss the primary research path of standard particle swarm algorithms. We describe each paper’s main considerations and contributions.

The seminal 1995 PSO paper by Kennedy and Eberhart [57] provided an introduction to the concept of optimization of nonlinear functions using swarms. The authors highlight simulation of social groups (such as bird flocking) as the motivation for the algorithm. The paper discusses research efforts to simplify the algorithm as much as possible. For example, as birds move in a flock, initial versions of the algorithm had programmed ‘craziness’ behavior and also nearest neighbor velocity matching. The PSO algorithm appeared to perform as well and as quickly without

these terms, and so they were removed (though the visual ‘flockiness’ of the swarm degraded as it evolved). A main thrust of this paper was to generalize the algorithm from 2-dimensional spaces to K -dimensional spaces. Technical results on this initial generalization were presented and proved promising. The initial version of PSO in this paper did not involve the velocity update constants ω , c_1 , and c_2 , though there is some discussion regarding the use of similar quantities. The authors did use a scaling factor of 2 in place of c_1 and c_2 and highlighted that efforts to discover whether there are general and optimal values of these constants, or if ideal values of these constants are problem specific, will be the subject of future research.

Eberhart and Kennedy (1995) [37] published a second paper on PSO. The focus of this paper was the introduction of a new swarm communication paradigm. While their previous paper focused on the global best swarm position $\mathbf{g}_{\text{best}}$ for swarm communication (whereby all particles can communicate their positions to each other at each iteration), this paper introduced locally-oriented communication via the $\mathbf{l}_{\text{best},i}$ quantity, which represents the best known position, up to the current iteration of the algorithm, among all particles in the neighborhood of particle i . In the $\mathbf{l}_{\text{best},i}$ communication paradigm, each particle has a local topological neighborhood (a subset of the swarm) within which they can communicate their fitness. In this paper the local communication neighborhood of particle i was defined literally as the nearest (in Euclidean distance) group of n_b particles. Note that $\mathbf{l}_{\text{best},i}$ replaces $\mathbf{g}_{\text{best}}$ in the update equations for this algorithm. The authors studied the effect of neighborhood sizes $n_b = 2$ and 6 on PSO performance. Initial observations in this study indicated that PSO with $\mathbf{l}_{\text{best},i}$ communication was more robust to entrapment in local optima than $\mathbf{g}_{\text{best}}$ communication. However, the $\mathbf{g}_{\text{best}}$ communication algorithm was quicker (in number of iterations) to converge. Additionally, this paper provided some discussion and comparison of PSO to genetic algorithms and indicated that PSO appears to

solve similar problems without some of the drawbacks and difficulties that genetic algorithms encounter—primarily PSO maintains a historical memory of the swarm via $\mathbf{g}_{\text{best}}$ whereas genetic algorithms lack such memory and a change in genetic populations destroys all past information gained during the algorithm regarding historical best fitness.

A refinement of the $\mathbf{l}_{\text{best},i}$ communication topology was provided by Kennedy (1997) [54]. In this paper he discontinues the idea of close spatial proximity within the search space to define communication neighborhoods. He defines the communication topology for particle i as the two particles with which it is adjacent to in the data array, that is, a local communication neighborhood is given by particle indices $\{i-1, i, i+1\}$. This is the so-called circle or ring communication topology. This paper also introduced the idea of velocity limiting—that is, if the velocity of the particle exceeds some threshold v_{max} in any dimension, then the particle’s velocity is set to v_{max} before displacement. Further, the author provided a detailed study of the effect of different values of v_{max} , as well as c_1 and c_2 , on PSO performance in optimizing network weights. It was observed that if v_{max} was set too small then the algorithm was susceptible to entrapment in a local optima (because particles take small steps in this case). This paper also presented a performance comparison of social-only and cognitive-only velocity update equations.

A more detailed study on communication topologies was provided by Kennedy (1999) [55]. The author studied four benchmark functions with PSO variants using different communication network structures. This study reinforced the contemporary conventional wisdom that the PSO variant using the $\mathbf{g}_{\text{best}}$ communications structure will converge to solution faster, but also be more susceptible to entrapment in local optima. The author introduced and studied four types of communication topologies:

1. stars ($\mathbf{g}_{\text{best}}$ communication topology),

2. wheels: one individual is connected to all others, but the others are only connected to that one,
3. circles or rings: of size n_b , each individual is connected to the n_b immediate neighbors in the data array,
4. random edges: n_b random symmetric connections are assigned between particle pairs.

We present graphs showing examples of each of these communication topologies for a swarm of $S = 10$ particles in Figure 1.4. In this paper the authors designed and implemented a simulation study so it could be analyzed via an ANOVA. The test factors of the study were: objective function (e.g., Rastrigin, Beale, etc.) and communication topology. This enabled them to analyze performance of PSO as a function of the objective, the communication topology, and their interaction. They observed that the test function main effect was not statistically significant, nor was the neighborhood type main effect. However, the interaction of function type and neighborhood type was significant. It was observed that the circle topology performed better than the wheel topology for three of the four test functions. The performance of these communication topologies was reversed for the Rastrigin function. The primary conclusion of the paper was that communication topology can impact performance and this impact may be dependent on the objective function. The question of which communication topology, $\mathbf{g}_{\text{best}}$ or $\mathbf{l}_{\text{best}}$, is ‘best’ was later revisited by Engelbrecht (2013) [40]. The results of this paper challenged the conventional wisdom that $\mathbf{g}_{\text{best}}$ may converge quicker but be more susceptible to getting trapped in local optima as compared to the $\mathbf{l}_{\text{best}}$ topology. Engelbrecht’s conclusion was that the best communication topology is dependent on the objective function.

Shi and Eberhart (1998) [84] published an advancement to PSO where they introduce the inertia weight parameter ω , and also introduced notation c_1 and c_2 for the weights of the social and cognitive components of the update equation, respectively. The authors demonstrated (through applications of PSO on Schaffer’s f6 benchmark function) that an inertia weight in the range $[0.9, 1.2]$ had a positive impact on the ability of PSO to find the global optimum. The authors observed that, especially at the early iterations of the algorithm, high values of ω encourage global searches while small values encourage local search. These authors provided another publication in 1998 which compared PSO to genetic algorithms [38]. They highlighted that this comparison to genetic algorithms is what led to the development of the inertia weight ω in PSO. Shi and Eberhart (1998) [85] provide an additional study on the impacts of ω and $v_{\max}$ and also provide some guidance on how to choose their values. They highlight that ω impacts PSO in providing a tradeoff between global (large values) and local (smaller values) exploration. The authors again use Schaffer’s f6 benchmark function and study a range of parameters. The author’s concluded that, in the absence of problem-specific knowledge, setting $v_{\max}$ to half the distance of the search space (if the space is a hypercube) is reasonable; and then values of $\omega \approx 0.8$ tend to have positive impact on the ability of PSO to quickly find the global optimum.

Study of the inertia weight ω and the constriction factors c_1 and c_2 remained a research topic for the next few years with results coming from Clerc (1999) [27] and Eberhart and Shi (2000) [39]. Clerc demonstrated that a constriction factor around the entire velocity update equation may be required to ensure convergence of the PSO algorithm [27]. His method was studied in detail on a number of optimization test functions by Eberhart and Shi (2000) [39]. To be concise, the authors parametrized the algorithm so that, in all cases, $\omega = 0.729$ and $c_1 = c_2 = 1.49445$. The primary

observation of this paper was that these values of the parameters enabled PSO to perform well in conjunction with $v_{\max}$ set equal to half the length of the search space (in case of a hypercube). Clerc and Kennedy (2002) [29] published an analysis that leads to the mathematics from which the constriction factors $\omega = 0.729$ and $c_1 = c_2 = 1.49445$ may be deduced. Further discussion of all developments in PSO up to 2001 can be found in Eberhart and Shi [36].

In 2006 Springer published a handbook of nature-inspired algorithms that included a chapter by Kennedy on PSO [56]. In this reference the author presented the canonical version of PSO as the current state of the art. He jokingly suggested that the values of the constriction factors (as used in Algorithm 1.1) are chosen because “they just work.” It is noted that if PSO employs these factors then the use of $v_{\max}$ has been demonstrated to be optional. The author introduced for the first time the idea of using a Gaussian function, instead of uniform distributions, in the velocity update equation, though this wasn’t further researched until 2013. The author also highlighted that further research is needed into communication topologies and how to determine best structures in general. There is also a short summary of approaches to hybridize PSO with other search algorithms.

By 2007, the depth and breadth of both the theoretical and empirical development of PSO, as well as its numerous applications, was so large that it led Bratton and Kennedy (2007) to publish a paper calling for a standard PSO (SPSO) algorithm [19]. The intention of these authors was to codify state-of-the-art developments of PSO to this point into an algorithm that the research community recognized as the standard and against which all future potential improvements should be benchmarked. This paper favored:

1. a local ring communication topology,

2. updated equations with constriction coefficient values as applied in Algorithm 1.1,
3. $S = 50$ particles.
4. non-uniform swarm initialization, and
5. a ‘let them fly’ boundary strategy wherein particles can move outside the search space, but in this case their fitness values will not be evaluated.

It is not clear if this version of the algorithm was widely adopted as an SPSO.

Another dimension of the PSO algorithm that garnered some research is the method of updating the swarm knowledge: whether each particle evaluates $\mathbf{p}_{\text{best},i}$ and $\mathbf{g}_{\text{best}}$ and then immediately updates its velocity (this is referred to as asynchronous updating [async]) -or- is it advantageous to evaluate the fitness of all particles first, and then to update velocities after the fitness of all particles are learned at their last positions (this is called synchronous updating [sync])? Rada-Vilela et. al. (2011) [77] provide a detailed study on the differences in performance of PSO using sync vs. async and crossed with $\mathbf{l}_{\text{best}}$ vs. $\mathbf{g}_{\text{best}}$ communication topologies. They appealed to ten well-known test functions and their study provides support that sync updating yields better results both in terms of successfully finding the optimum of f as well as in convergence rate. The authors indicate that this finding contradicts the previous conventional wisdom within the community on this topic. Dongmei Wu (2014) [95] reported an adjustment to async PSO which employs adaptive switching to the inertia weight ω and this adjustment was seen to bring performance of async PSO in line with sync PSO.

Community efforts to define and adopt standard PSO algorithms continued after Kennedy’s 2007 paper on the subject. Clerc (2012) [28] published a technical report

at the HAL open archives which details all currently acknowledged SPSO algorithms. We discuss the technical details presented in this report in the next sections.

1.5 PSO Algorithm and Standard Variants

1.5.1 Canonical PSO

The canonical PSO algorithm as recognized by the community appears to be close to that presented by Bratton and Kennedy (2007) [20]. This is based on the observation that canonical PSO is programmed in the R hydroPSO package [98], which is currently maintained by some of the main researchers in the PSO field, as described below. The aspects of this algorithm include:

1. $S = 40$ particles,
2. use of Bratton and Kennedy's inertia, cognitive, and social weights in the velocity update equation: $\omega = 0.72984, c_1 = c_2 = 2.05\omega = 1.496172$,
3. particle velocities at initialization are computed as $\mathbf{v}_i \sim U_K(\mathbf{l}_b - \mathbf{x}_i, \mathbf{u}_b - \mathbf{x}_i)$,
4. use of the $\mathbf{l}_{\text{best}}$ communication topology with an average of $n_b = 3$ neighbors per particle. A random initialization of neighborhood links occurs in the event that $f(\mathbf{g}_{\text{best}})$ does not improve within an iteration (this approach is called adaptive random topology), and
5. absorbing wall particle confinement.

1.5.2 SPSO 2007

The HAL technical manual on SPSO by Clerc [28] discusses an SPSO 2006; however, it is similar to SPSO 2007 in most respects except for employing the adaptive random $\mathbf{l}_{\text{best}}$ communication topology and use of async updating. The major aspects of SPSO 2007 include:

1. $S = 10 + \lfloor (2 * \sqrt{K}) \rfloor$ particles,
2. a refinement on the velocity update weight parameters: $\omega = \frac{1}{2\ln(2)} \approx 0.721$, $c_1 = c_2 = 2.05 * \omega = \frac{1}{2} + \ln(2) \approx 1.193$,
3. particle velocities at initialization are computed as $\mathbf{v}_i \sim U_K\left(\frac{\mathbf{l}_b - \mathbf{x}_i}{2}, \frac{\mathbf{u}_b - \mathbf{x}_i}{2}\right)$,
4. use of the adaptive random $\mathbf{l}_{\text{best}}$ communication topology with an average of $n_b = 3$ neighbors per particle,
5. absorbing wall particle confinement,
6. if $\mathbf{l}_{\text{best}} = \mathbf{p}_{\text{best},i}$, then the cognitive term of the velocity update equation is removed for particle i , and
7. sync updating is employed, and the indices of the `for` loop are randomly permuted each time this loop is entered to move the particles.

1.5.3 SPSO 2011

Currently it appears that Zambrano-Bigiarini, Clerc, and Rojas-Mujica are the contemporary leaders in holding and advancing the SPSO algorithm and providing general software for its implementation and algorithm benchmarking [96–98]. In their 2011 IEEE paper [97], they published a major update to the structure of the PSO algorithm. In simulation studies they observed a bias in PSO if the solution was on the boundary of $\mathcal{X}$ and attributed this to the fact that the random perturbations to the update equations were drawn from a multivariate uniform distribution and updating was carried out *dimension by dimension* (and so were parallel to the axes of the search space). Thus, the primary change to SPSO in 2011 was a decoupling of the velocity update equations with respect to their orientation to the axes of the search space. To do this they defined the following points in K -space: a point “a bit

beyond” the best known position

$$\mathbf{p}_i^*(t) = \mathbf{x}_i(t) + c_1 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{p}_{\text{best},i} - \mathbf{x}_i(t)),$$

and a point “a bit beyond” the neighborhoods current best position

$$\mathbf{l}_i^*(t) = \mathbf{x}_i(t) + c_2 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{l}_{\text{best},i} - \mathbf{x}_i(t)).$$

The center of gravity of these points and the current particle position is

$$\mathbf{q}_i(t) = \frac{\mathbf{x}_i(t) + \mathbf{p}_i^*(t) + \mathbf{l}_i^*(t)}{3}. \quad (1.6)$$

Now, instead of drawing a random point from a rectangular distribution (as in the standard update equations), a random point is drawn from a hypersphere

$$\mathbf{x}_i^* \sim H_n(\mathbf{q}_i(t), \|\mathbf{q}_i(t) - \mathbf{x}_i(t)\|),$$

where the random sphere H_n is centered at $\mathbf{q}_i(t)$ and has radius $\|\mathbf{q}_i(t) - \mathbf{x}_i(t)\|$. Note that $\|\mathbf{x}\| := \sqrt{x_1^2 + \dots + x_n^2}$ is the Euclidean norm. Finally, the velocity update equation is defined as

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + \mathbf{x}_i^* - \mathbf{x}_i(t), \quad (1.7)$$

and the particle position update is still $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$. If $\mathbf{l}_{\text{best}} = \mathbf{p}_{\text{best},i}$, then the social factor is dropped (similar to SPSO 2007) and the center of gravity is computed as

$$\mathbf{q}_i(t) = \frac{\mathbf{x}_i(t) + \mathbf{p}_i^*(t)}{2}.$$

Velocity initialization is the same as in canonical SPSO: $\mathbf{v}_i \sim U_K(\mathbf{l}_b - \mathbf{x}_i, \mathbf{u}_b - \mathbf{x}_i)$. SPSO 2011 uses the same updating scheme as SPSO 2007: use of the adaptive random $\mathbf{l}_{\text{best}}$ communication topology with an average of $n_b = 3$ neighbors per particle; and sync updating is employed where the indices of the **for** loop are randomly permuted each time this loop is entered to move the particles. Last, for particle confinement, instead of an absorbing wall where particle velocities are set to 0, SPSO 2011 implements a reflecting wall (particles that hit the boundary of the search space literally bounce back) as described in Algorithm 1.3.

Algorithm 1.3: confine: Reflecting wall particle confinement.

- 1: **Input:** Particle i 's position $\mathbf{x}_i$
- 2: **Output:** A valid particle position in $\mathcal{X}$ and an adjustment to the particle's velocity $\mathbf{v}_i$.
- 3: // Check each dimension independently
- 4: **if** $\{x_{ik} \notin (l_j, u_k)\}$
- 5: $\{v_{ik} \leftarrow -0.5v_{ik}\}$
- 6: **if** $\{x_{ik} < l_k\}; \{x_{ik} \leftarrow l_k\}$ **endif**
- 7: **if** $\{x_{ik} > u_k\}; \{x_{ik} \leftarrow u_k\}$ **endif**
- 8: **endif**
- 9: **return**

1.6 Statistics: Optimization Problems and Possible Applications of PSO

In this section we introduce some basic problem spaces in statistics where PSO may prove to be a useful solution paradigm. Two main classes of statistical problems which involve optimization include parameter estimation techniques (including method of least-squares and maximum likelihood [ML] estimation) and optimal design of experiments. We illustrate the application of PSO to solve toy examples of each of these problems in the next subsections.

1.6.1 Parameter Estimation

The method of least-squares model fitting grew from the fields of astronomy and geodesy in the eighteenth century and was motivated by problems dealing with navigation. A number of individuals contributed to the early development of the method. Of note is a seminal contribution of Pierre-Simon Laplace in 1799 who attempted to specify a probability distribution on the errors [48]. He assumed a two-sided exponential distribution which later became known as the Laplace distribution. He used the absolute deviation of errors as the objective, and while he hoped the solution would be the arithmetic mean, it turned out that this set of assumptions led to the median as a solution. In 1805 Adrien-Marie Legendre was the first researcher to publish a concise formulation of the method of least squares as an algebraic fitting procedure. Legendre's formulation of the problem immediately gained recognition by contemporary leading astronomers and geodesists. Legendre's method was later justified from a statistical basis by Carl Friedrich Gauss and Laplace. In 1809 Gauss published his method for calculating the orbits of celestial bodies in which he connected the method of least squares with probability principles that led to the development of the Gaussian (normal) distribution [48].

Today we understand the approach more abstractly as described in classical statistical theory. Specifically, the exponential family of distributions has a number of nice and well known mathematical and statistical properties [12]. Among these is the general mathematical tractability of ML estimates and, for the Gaussian distribution as a specific example, the equivalence of the ML estimate (MLE) and method of least-squares approach to parameter estimation. However, if one strays a bit from the exponential family, then mathematical tractability of parameter estimates from either approach can be lost. Specifically, non-linear models can have multimodal least-squares objective functions or likelihood functions, and solving the parameter

estimation problem for these models may require highly detailed mathematical solutions, numerical optimization, and often some knowledge of the true solution to ensure success of the solution approach [83].

1.6.1.1 Non-linear Models We consider a simple example of nonlinear models presented in Seber and Wild (2003) [83] pp. 92. The non-linear least squares problem is formulated as follows. The setting is a set of observed response data $\mathbf{y} : n \times 1$ with covariates $\mathbf{x} : n \times 1$ related by a function f parametrized by a vector $\boldsymbol{\theta} : p \times 1$. Specifically

$$\mathbf{y} = f(\mathbf{x}, \boldsymbol{\theta}) + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon}$ represents a vector of errors and we currently place no probabilistic assumptions on this quantity. Parameter estimation can be stated as the optimization of some objective function $h(\boldsymbol{\theta})$, such as the sum-of-squared error (SSE) terms $h(\boldsymbol{\theta}) = \|\mathbf{y} - f(\mathbf{x}, \boldsymbol{\theta})\|^2$, as follows:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} h(\boldsymbol{\theta}). \quad (1.8)$$

As analytical solutions are often intractable for nonlinear models f [83], a common approach to this optimization is to apply (some variant) of Newton's method—an iterative method that considers a second-order Taylor expansion of $h(\boldsymbol{\theta})$ and constructs a sequence of estimates $\{\hat{\boldsymbol{\theta}}^0, \hat{\boldsymbol{\theta}}^1, \dots, \hat{\boldsymbol{\theta}}^t\}$ which converges to a solution (not guaranteed to be the global minimum). Because Newton's method approximates the objective by a quadratic function via a Taylor expansion, these algorithms will work well when the underlying objective is truly quadratic. Further, since any function may be approximated by a quadratic function in a small enough neighborhood, it is beneficial and sometimes even necessary that the initial guess of the solution $\boldsymbol{\theta}^0$ be

close to the true value in order to enable a successful optimization. The details of the Newton-Raphson algorithm are as follows [11]. Denote the optimizer of $h(\boldsymbol{\theta})$ by $\hat{\boldsymbol{\theta}}$. Consider the objective function evaluated at this value $h(\hat{\boldsymbol{\theta}})$, and take the second order Taylor expansion around a guess $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_i$ to get

$$h(\hat{\boldsymbol{\theta}}) \approx h(\hat{\boldsymbol{\theta}}_i) + (\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_i)' \mathbf{g}_{\hat{\boldsymbol{\theta}}_i} + \frac{1}{2}(\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_i)' \mathbf{H}_{\hat{\boldsymbol{\theta}}_i} (\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_i), \quad (1.9)$$

where

$$\mathbf{g}_{\hat{\boldsymbol{\theta}}_i} = \left. \frac{\partial h(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_i} \quad \text{and} \quad \mathbf{H}_{\hat{\boldsymbol{\theta}}_i} = \left. \frac{\partial^2 h(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \otimes \partial \boldsymbol{\theta}'} \right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_i}$$

are the gradient vector and Hessian matrix, respectively, evaluated at the guess $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_i$, and (only here in this dissertation) $\otimes$ represents the Kronecker product. To find the vector $\hat{\boldsymbol{\theta}}$ that minimizes $h(\boldsymbol{\theta})$, take the derivative of $h(\hat{\boldsymbol{\theta}})$ as in Eq. 1.9, set it equal to $\mathbf{0}$ to find the stationary point, and solve for $\hat{\boldsymbol{\theta}}$. This yields

$$\frac{\partial h(\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}} = \mathbf{0} \implies \mathbf{g}_{\hat{\boldsymbol{\theta}}_i} + \mathbf{H}_{\hat{\boldsymbol{\theta}}_i} (\hat{\boldsymbol{\theta}} - \hat{\boldsymbol{\theta}}_i) = \mathbf{0}. \quad (1.10)$$

Last, the update equation to the algorithm is given by solving Eq. (1.10) for $\hat{\boldsymbol{\theta}}$, which gives the iteration

$$\hat{\boldsymbol{\theta}}_{i+1} = \hat{\boldsymbol{\theta}}_i - \mathbf{H}_{\hat{\boldsymbol{\theta}}_i}^{-1} \mathbf{g}_{\hat{\boldsymbol{\theta}}_i}. \quad (1.11)$$

Eq. (1.11) thus will update the estimate of the minimum until convergence.

This approach works well as long as the initial guess is not too far from the global optima. It is possible that the algorithm could be entrapped in a local optima since at such a point we have $\mathbf{g}_{\hat{\boldsymbol{\theta}}} = \mathbf{0}$, and so further iterations do not change the estimate. To demonstrate this we consider a simple exponential decay model with

parameter vector $\boldsymbol{\theta} = (\alpha, \beta)$:

$$y = \alpha \exp\{-\beta x\} + \epsilon. \quad (1.12)$$

This model may of course be linearized via $\ln(y)$, but we will illustrate the complexities that nonlinear functions introduce to the least squares solution paradigm by keeping with the scale y . A hypothetical sample of size $n = 4$ was taken from this model, and the resulting data were $\{(x_i, y_i)\} = \{(-2, 0), (-1, 1), (1, -0.9), (2, 0)\}$. The least-squares objective is

$$h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2.$$

We present a contour map of h for these data in Figure 1.5. It is readily seen that h has a global minimum near $\boldsymbol{\theta} = (0.090, 0.600)$ and a local minimum near $\boldsymbol{\theta} = (-0.063, -0.699)$. Thus the researcher, before implementing a Newton-type search, requires some knowledge of the behavior of the objective function in order to ensure that the optimization routine will converge to the global minimum.

While we can inspect this objective function via a contour map in 2-D, higher dimensional objective functions may not be so easy to investigate. We utilized R's built in optimization routines, specifically `nlm`, to illustrate how different starting values affect the success of the Newton-type algorithm to find the global minimum. We ran four sequences of the algorithm, each with different starting values. The results of the search algorithms are presented in Figure 1.6. Each run of the Newton algorithm is indicated by a different color and plot symbol. The starting point of each algorithm is indicated by a black point. We can see that the blue and green chains were initialized at starting values that were in the domain of attraction of the global minimum, while the red and purple chains were initialized at starting values

Table 1.1: Locations and objective function values of the local and global minimum of $h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$.

optimum	$\hat{\alpha}$	$\hat{\beta}$	$h(\hat{\boldsymbol{\theta}} = \hat{\alpha}, \hat{\beta})$
local min	-0.063	-0.699	1.727
global min	0.087	0.620	1.690

in the domain of attraction of the local minimum. We summarize the locations and SSE values in Table 1.1. To illustrate how each solution (local or global minimum) affects the fitted model, we present a scatter plot of the data and each of the fitted models in Figure 1.7.

Next, we implemented SPSO 2007 to optimize $h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$. The algorithm converged to the global minimum at 54 iterations using only $S = 12$ particles. We present contour plots showing the swarm state at six different time snippets of the algorithm in Figure 1.8. These results are encouraging in that, where classical Newton-type algorithms require additional information (such as some knowledge of the true global optima, or perhaps multiple runs of the algorithm), a single run of PSO with a small number of particles is able to find the global optimum in a small number of iterations. A partial reason for this is that PSO by default is initialized using a set of randomly chosen starting points.

1.6.1.2 Mixture Distributions Parameter estimation for mixture distributions is another problem space where the objective function may be multimodal. A mixture distribution is formulated as follows. Let random variable $X \sim g(\boldsymbol{\phi})$, where parameter vector $\boldsymbol{\phi} = (\boldsymbol{\pi}', \boldsymbol{\theta}'_1, \dots, \boldsymbol{\theta}'_N)'$, weight vector $\boldsymbol{\pi} = \{\pi_j\}$, and the density g is a weighted

average of a set of densities f from the same parametric family [63]. Specifically,

$$g(x|\boldsymbol{\phi}) = \sum_{j=1}^N \pi_j f(x|\boldsymbol{\theta}_j) \quad 0 \leq \pi_j \leq 1 \quad \forall j, \quad \sum_{j=1}^N \pi_j = 1. \quad (1.13)$$

Given a random sample $\mathbf{x} : n \times 1$, the likelihood function is

$$L(\boldsymbol{\phi}|\mathbf{x}) = \prod_{i=1}^n g(x_i|\boldsymbol{\phi}) = \prod_{i=1}^n \sum_{j=1}^N \pi_j f(x_i|\boldsymbol{\theta}_j). \quad (1.14)$$

The likelihood in Eq. (1.14) is a high-dimensional multimodal function of $\boldsymbol{\phi}$. This problem is a typical example used to illustrate the use of the Expectation-Maximization (EM) algorithm. Another approach to estimating the parameters that has been developed in recent decades is Bayesian Data-Augmentation (DA). Both EM and DA are *latent variable* approaches wherein a multinomial random variable Z is introduced in the modeling but is unobserved in the data. Z encodes the class membership of an observation. If $Z = j$ is known, then we have improved our knowledge from $X \sim g(\boldsymbol{\phi})$ to $X|Z = j \sim f(x|\boldsymbol{\theta}_j)$. Thus, a ‘complete’ data set would be $[\mathbf{x}, \mathbf{z}]$ where in-class knowledge for each x_i is known. EM and DA are similar in the sense that they ‘try out’ different $\mathbf{z}$ at each iteration, eventually find the optimal estimate of class membership probability, and then marginalize over $\mathbf{z}$ in order to help optimize the observed data likelihood in Eq. (1.14). The two algorithms differ as follows. EM is essentially a gradient search and so is not without the pitfalls of gradient searches, e.g., EM can get stuck in local optima or saddle points if the initial guess is not close enough to the global optima. However, provided the starting value is in the domain of attraction of the global optima, EM is guaranteed to converge to the ML estimate. Another drawback of EM is that there is no built-in uncertainty quantification (UQ). That is, EM only produces point estimates of the parameters and not their standard errors. DA is superior from the UQ perspective. Because

DA is a fully Bayesian approach, uncertainty estimation is built into the algorithm in the form of producing posterior distribution estimates on the parameters. DA, however, is computationally complex—it is a Monte-Carlo (MC) approach which requires prediction of the latent class variable Z for each datum and MC integration. If the model contains many mixture components, implementation of such algorithms can become quite complex, though, this has been mitigated some in recent years due to advancements in statistical computing [63].

We now describe the high-level formulation of the EM algorithm so that the reader can gain some intuition of the latent variable approach [12]. Let $L_C(\phi|\mathbf{x}, \mathbf{z})$ represent the complete data likelihood. Keep in mind, however, that the problem is to optimize the observed data likelihood $L(\phi|\mathbf{x})$. Let the current guess of the parameter be $\phi^{(t)}$, and define the quantity

$$Q(\phi, \phi^{(t)}, \mathbf{x}) = E_{\phi^{(t)}} (\log L_C(\phi|\mathbf{x}, \mathbf{z})) \quad (1.15)$$

$$= \int \log L_C(\phi|\mathbf{x}, \mathbf{z}) f_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x}, \phi^{(t)}) d\mathbf{z}, \quad (1.16)$$

where $f_{\mathbf{z}|\mathbf{x}}$ is the posterior predictive distribution for class memberships, conditional on the observed data $\mathbf{x}$ and the current parameter guess $\phi^{(t)}$. We can see that this objective Q involves taking an expectation in order to remove dependence of Q on the unobserved data $\mathbf{z}$. With Q defined, the basic steps of the EM algorithm are:

1. **Initialize:** Pick starting point $\phi^{(0)}$.
2. **E-Step:** Compute function $Q(\phi, \phi^{(t)}, \mathbf{x})$.
3. **M-step:** Compute $\phi^{(t+1)} = \operatorname{argmax}_{\phi} Q(\phi, \phi^{(t)}, \mathbf{x})$
4. **Iterate:** Until convergence.

Most of the complexity of implementing the EM algorithm lies in identifying and computing all terms in Q .

We illustrate application of the EM algorithm to a mixture distribution problem, a simple textbook example as described in [63]. Consider a mixture of two Gaussian distributions with unknown means $\phi = (\mu_1, \mu_2)$ but known and constant standard deviations $\sigma_1 = \sigma_2 = 1$. Thus the model is

$$g(\phi) = \pi \mathcal{N}(\mu_1, 1) + (1 - \pi) \mathcal{N}(\mu_2, 1).$$

Additionally, we will assume that $\pi = 0.7$ is known. We simulated $n = 500$ observations with $\mu_1 = 0$ and $\mu_2 = 2.5$. We present a smoothed density of the sampled data in Figure 1.9.

A contour plot of the negative log-likelihood (which should be minimized) is shown in Figure 1.10. This plot illustrates that the likelihood surface for mixture distributions is multimodal with the global minimum occurring at true parameter value $\phi = (0, 2.5)$. Note that this is yet a simplified version of the problem because we have assumed known $\pi = 0.7$. In most mixture distribution estimation problems, the mixture proportion vector π is unknown and must be estimated. For this simplified Gaussian problem, we would find a symmetry in the log-likelihood function when $\pi = 0.3$, but with a matching global mode at point $\phi = (2.5, 0)$. This indicates an identifiability problem for estimation of mixture distribution parameters in general (e.g., labels could be switched ‘ X_1 ’ to ‘ X_2 ’). This is commonly dealt with after estimation by ordering computed parameter values. Nonetheless, this simple example is effective in pointing out that likelihood surfaces for mixture distributions are high-dimensional and multimodal.

We performed three runs of the EM algorithm on the log-likelihood function.

The results of these searches are presented in Figure 1.11. The true parameter value is indicated in red. The three chains were chosen with starting points in different domains of attraction. The blue chain reaches the global optima, the green chain terminates at a saddle point, and the purple chain terminates at a local optima. This illustrates some of the difficulties in implementing the EM algorithm on complicated optimization problems.

Now, while the EM algorithm works with the completed likelihood L_C and then, during iteration, calculates the observed likelihood L , there is nothing that says one cannot optimize L directly in order to retrieve the ML estimates. PSO should be able to find MLEs for mixture distributions with ease, and, noting that EM does not provide standard errors, we believe that PSO could be viewed as a superior approach to the EM algorithm for estimating parameters of mixture distributions as it should be able to produce the same information as EM while mitigating some of the drawbacks (e.g., sensitivity of solution to starting value).

We applied SPSO 2011 to the two-component Gaussian mixture problem. Recall that SPSO 2011 uses a sync updating scheme on $S = 40$ particles and also uses the spherical velocity update equations. The algorithm converged in 80 iterations to $\hat{\phi} = (0.090, 2.43)$, which is the same solution returned by the one successful EM run. We present 6 of the 80 iterations in Figure 1.12. Note that, in the early iterations, the particles may be outside the plot region because we initialized PSO particles on the ranges $[-10, 10]$ in each direction. The wide initialization range for PSO reflects our large uncertainty in the locality of the MLE's in contrast with EM, which requires a close initial guess. In total, these results speak favorably to the promise of PSO to become a mainstay algorithm for parameter estimation in mixture distribution problems.

1.6.2 Optimal Design of Experiments

Optimal design of experiments (DoE; sometimes referred to more generally as computer-aided design) is another problem class in statistics where PSO has the potential to make major contributions. As subsequent chapters of this dissertation will focus on developing and implementing PSO on DoE problems, we will not provide toy examples here. Rather, this subsection will serve as a literature review since PSO has already found some applications for DoE in the statistics literature.

Optimal DoE is concerned with determining at what settings to take experimental runs (i.e., what specific values of the experimental factors should be tested), and this determination is made by appealing to an objective function which gives concrete meaning to ‘optimal design’. Consider a fixed number N experimental runs on K experimental factors, and say we want that design $\mathbf{X} : N \times K$ which, in conjunction with the model and corresponding model matrix $\mathbf{F} : N \times p$, minimizes the variance $\text{Var}(\hat{\beta}) = \sigma^2(\mathbf{F}'\mathbf{F})^{-1}$ of the parameter $\beta : p \times 1$. Since σ^2 conveys information only about the size of errors on the measurement scale, and further since $(\mathbf{F}'\mathbf{F})^{-1}$ is a matrix, it is customary to use the objective $D = |(\mathbf{F}'\mathbf{F})^{-1}|$ and to seek the design $\mathbf{X}$ which minimizes D . There are numerous such criteria, each with a different purpose. We will expand in detail on these topics in the next chapter.

To date, applications of PSO in optimal DoE have been few but important. Chen et al. (2011) [23] provide a technical report which highlights that (up to this point) PSO as an optimization approach had not yet made it into the statistics literature. The authors propose variants of PSO to find optimal experimental designs for both linear and nonlinear models. They note that PSO can generate many types of optimal designs quickly, including under non-differentiable criterion such as minimax optimal designs where current approaches have failed.

Chen et al. (2013) [25] provide a contribution toward optimizing latin hypercube

designs. The authors highlight that as the number of design points gets large, the total number of Latin Hypercube Designs (LHDs) grows exponentially. This makes the search for the optimal LHD complicated. They propose a new algorithm called LaPSO as a customized approach for LHD optimal design search. They apply GPU (graphical processing unit) acceleration to aid the search and improve the speed of the algorithm. They demonstrate that LaPSO is more stable than existing approaches and capable of improving known results.

Chen et al. (2013) [26] offer another contribution focused on PSO for constructing uniform designs on irregular regions. This paper focuses on the Central Composite Discrepancy (CCD) measure for uniformity of a design over an irregular experimental region. Computing the CCD in this setting in an efficient way was until this point a challenge. The authors use GPU parallelization to improve speed. They illustrate the approach on several 2-D to 5-D benchmark problems.

Chen et al. (2014) [24] offer another contribution for PSO to compute minimax optimal designs. Minimax criterion focus on worst-case scenarios—this includes the G -optimal criteria (i.e., finding that design that minimizes the maximum prediction variance). The authors claim PSO is beneficial because such criterion can be difficult to compute, let alone optimize. They provide applications to linear and non-linear models and offer an augmented PSO algorithm that computes optima for saddle point criteria. We note that they focus on continuous designs.

Wong et al. (2015) [94] offer a contribution for application of PSO to find optimal designs for continuous designs in mixture experiments. Optimization in mixture experiments is a challenge because the design space is a K -simplex embedded in $\mathbb{R}^K$. They implemented a ‘let them fly’ approach and if the particles left the simplex, then before evaluating the objective function, they simply normalize the vector to unit length first (thereby restricting rows of the design matrix to stay on the simplex).

They were able to demonstrate that PSO is a candidate for design generation (on the continuous design paradigm) for mixture experiments by comparing PSO results to SAS OPTEX procedure and R's AlgDesign package [93].

Lukemire et al. (2018) [62] offer a contribution focusing on a modified PSO algorithm for finding D -optimal designs with both discrete and continuous factors but where the response is binary (e.g., logistic regression). Their primary contribution is that theoretical results rarely exist for such models so offering a robust algorithmic framework for such problems is an excellent contribution. They provide a robustness study of the algorithm to violations of the model assumptions.

Last, a recent contribution by Shi et al. [86] focuses on PSO for finding locally and Bayesian D -optimal designs. These authors also focus on continuous designs. They explore both linear and non-linear models. They also offer a set of web-based tools where practitioners can gain access to their algorithms.

1.7 Conclusions

In this chapter we introduced the reader to the basic Particle Swarm Optimization algorithm. We described all technical aspects of the algorithm and provided an in-depth literature review into the historical development of standard PSO. We intend to primarily use bPSO, SPSO 2007, and SPSO 2011 in further research into using PSO as an approach to solving optimization problems in statistics. Next, we provided a high level but thorough review of problem spaces in statistics that may benefit from adopting PSO as an approach to finding solution. These problem spaces include parameter estimation in non-linear models, parameter estimation for mixture distributions, and computer generated DoE (specifically optimal design). We illustrated the efficacy of PSO on toy problems in non-linear regression and mixture distributions and contrasted pros/cons against commonly applied optimization

routines—this included the Newton-Raphson and EM algorithm, respectively. Last, we concluded with a literature review of the few but important publications in statistics regarding applications of PSO to optimal DoE. Based on the review and information presented in this chapter, we believe further research into PSO in each problem space is warranted.

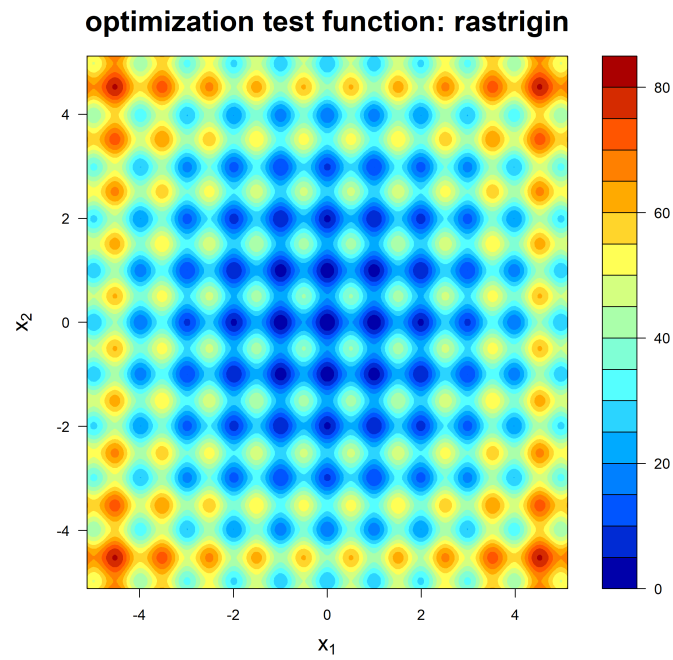


Figure 1.1: Contour plot of the Rastrigin function for $n = 2$.

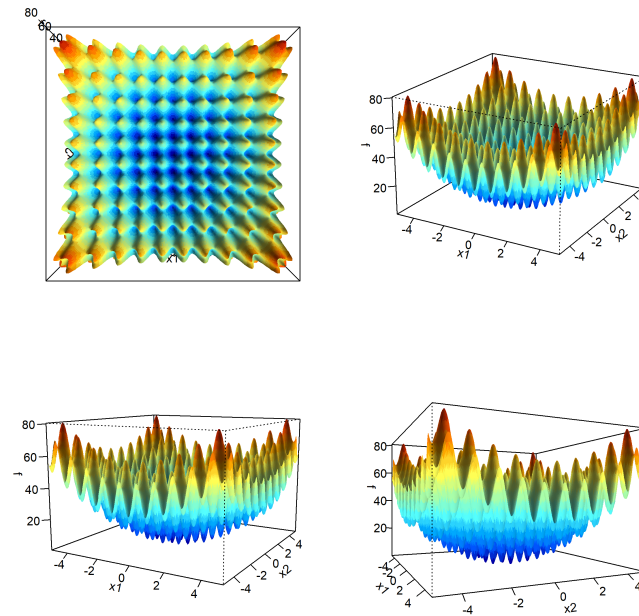
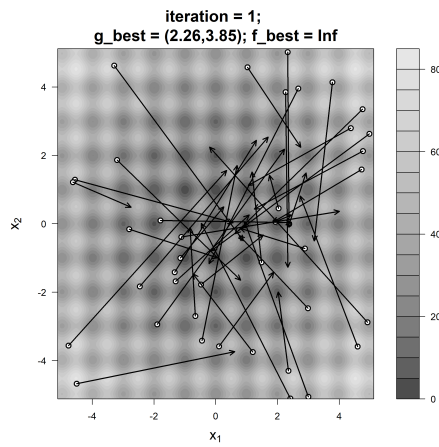
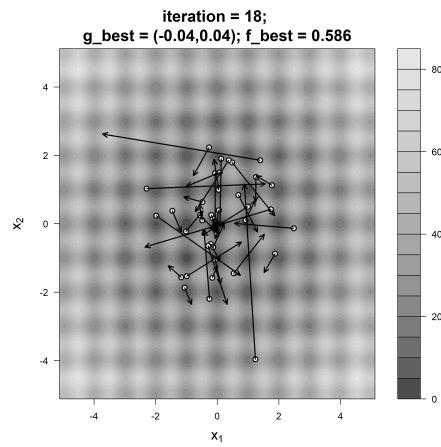


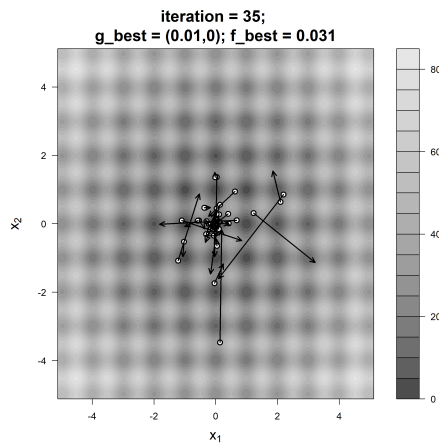
Figure 1.2: Perspective plot of the Rastrigin function for $n = 2$.



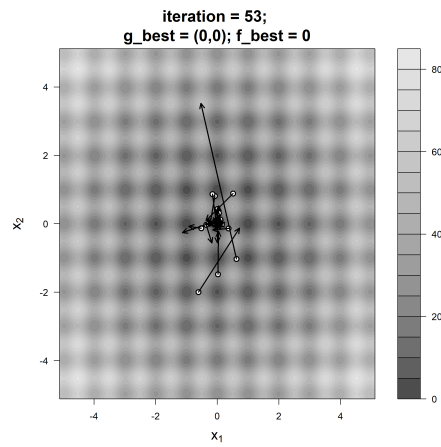
(a) Initialization



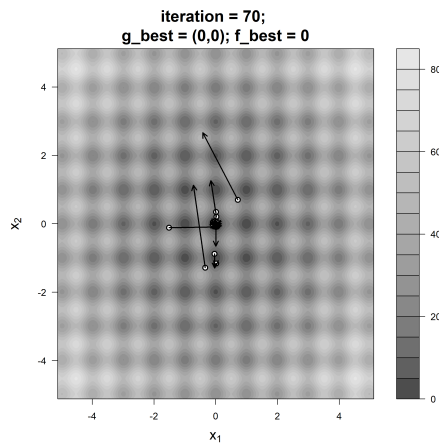
(b) Iteration 18



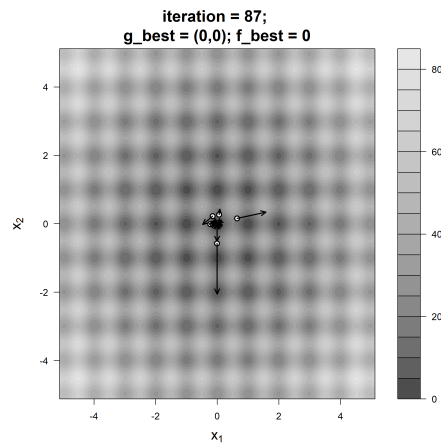
(c) Iteration 35



(d) Iteration 53



(e) Iteration 70



(f) Iteration 87

Figure 1.3: Application of bPSO to the Rastrigin function. The sequence of figures show the swarm state at the indicated iterations.

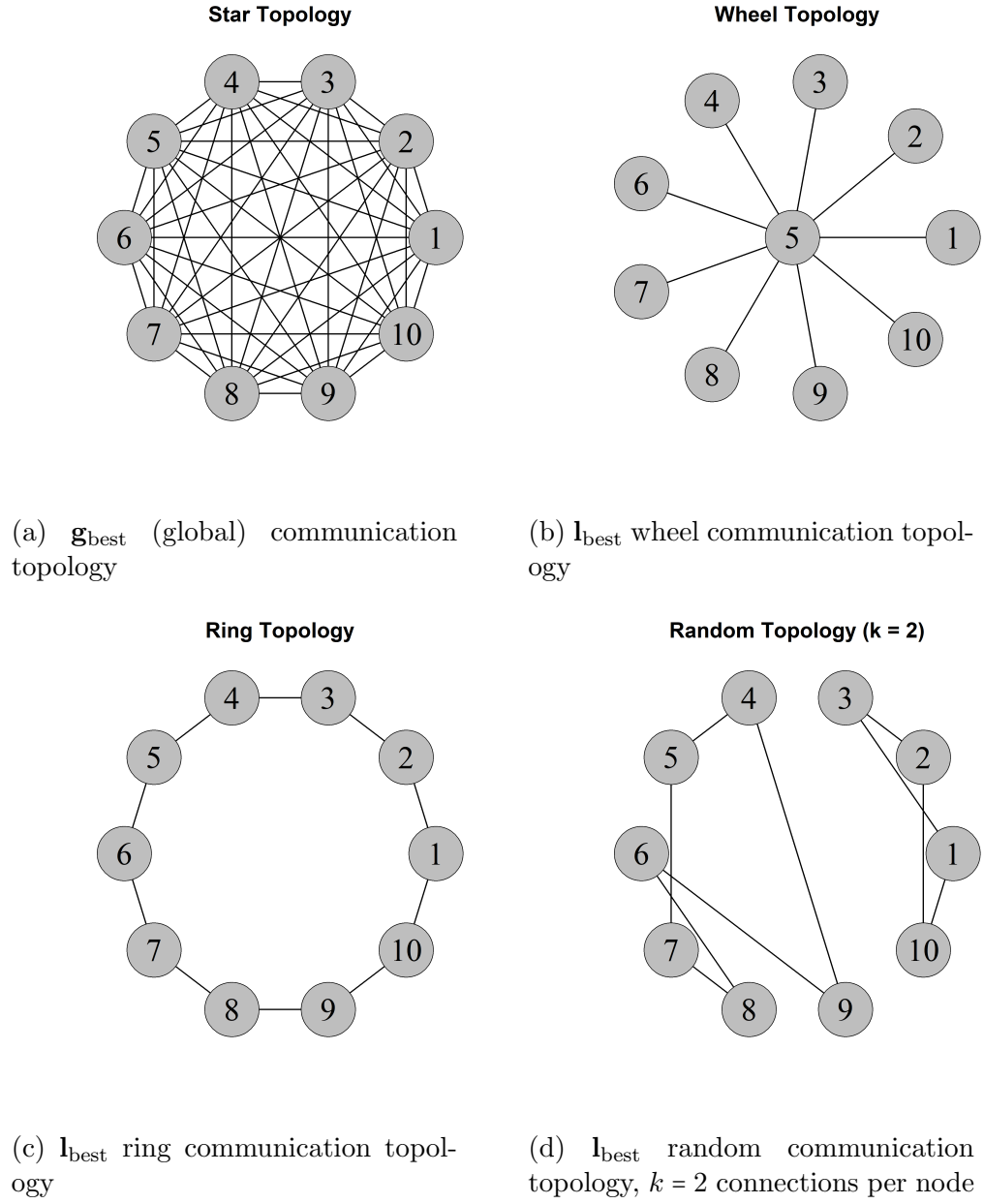


Figure 1.4: Example graphs illustrating different particle communication topologies for $S = 10$ particles. The labeling of the vertices represents the particle index.

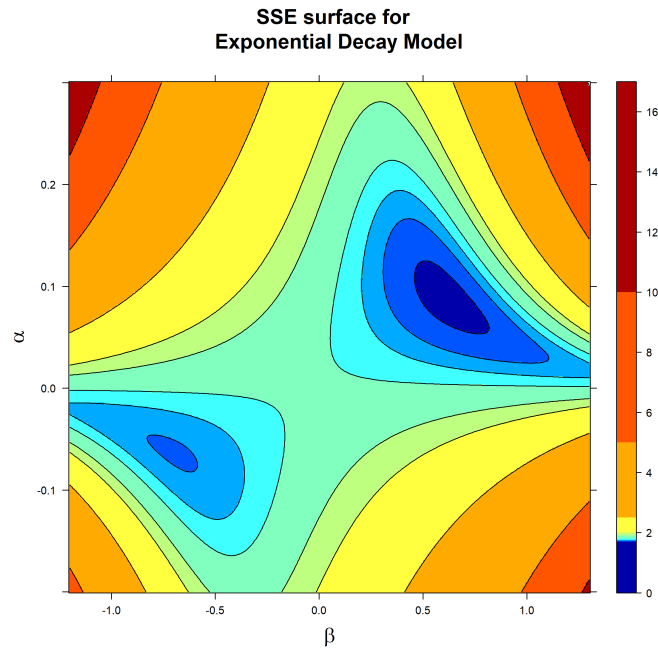


Figure 1.5: Contour plot of SSE: $h(\boldsymbol{\theta}) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$.

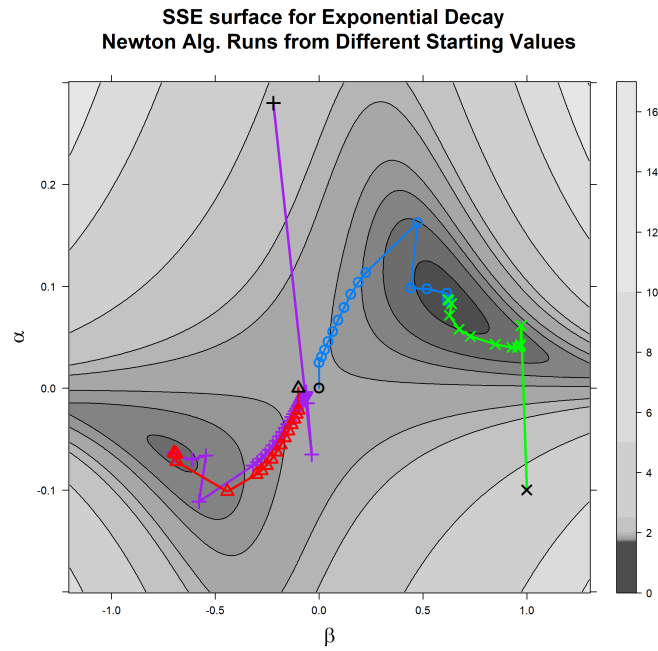


Figure 1.6: Four example sequences of a Newton-type optimization algorithm (represented by different colors). The black point indicates the starting value.

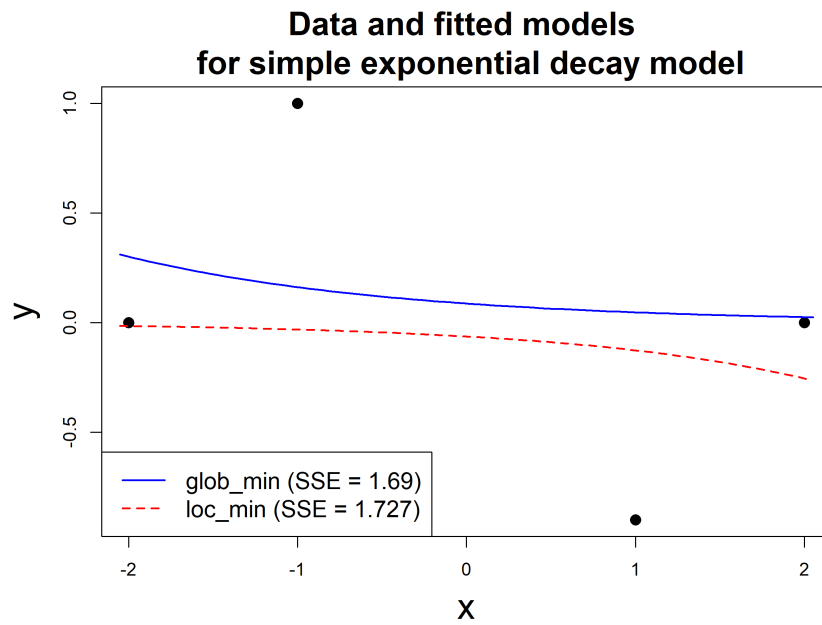
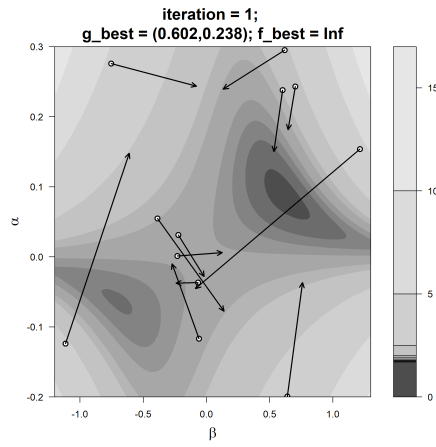
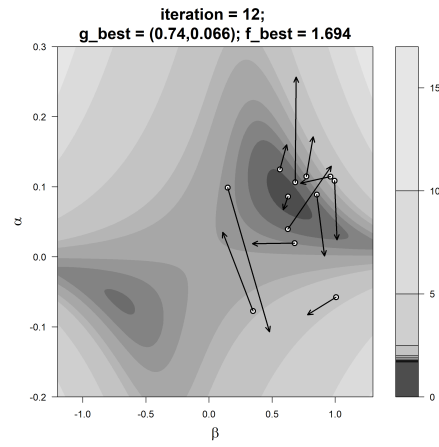


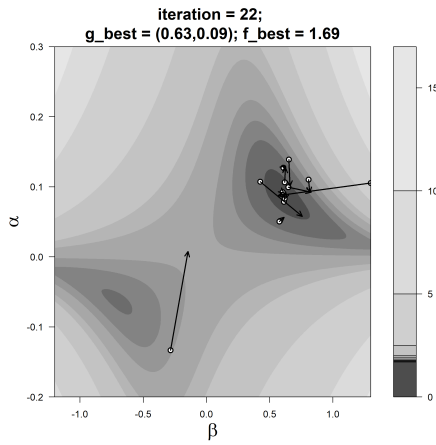
Figure 1.7: Scatterplot of the exponential decay model data and curves showing the fitted models for the global (blue) and local (red) minimum solutions.



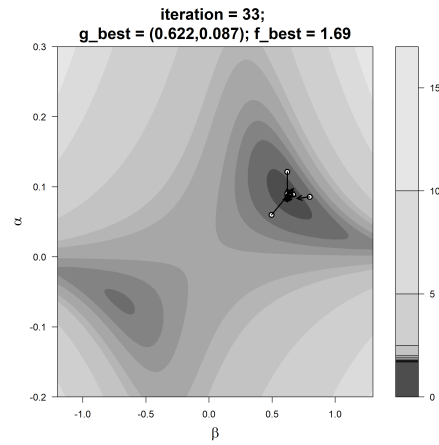
(a) Initialization



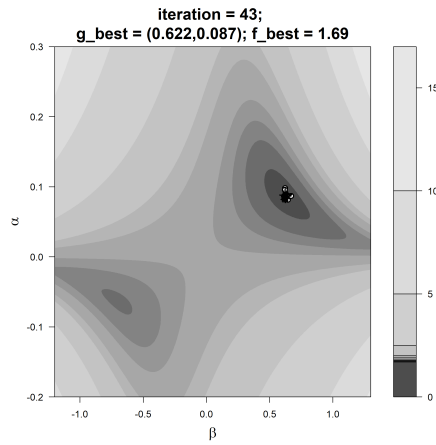
(b) Iteration 12



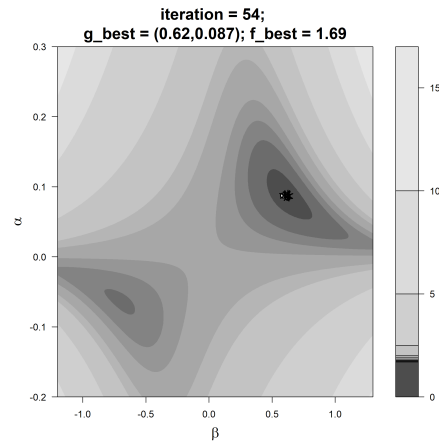
(c) Iteration 22



(d) Iteration 33



(e) Iteration 43



(f) Iteration 54

Figure 1.8: Application of SPSO 2007 to optimize $h(\theta) = \sum_{i=1}^4 (y_i - \alpha \exp\{-\beta x_i\})^2$. The algorithm converged to the global minimum at 54 iterations using only $S = 12$ particles.

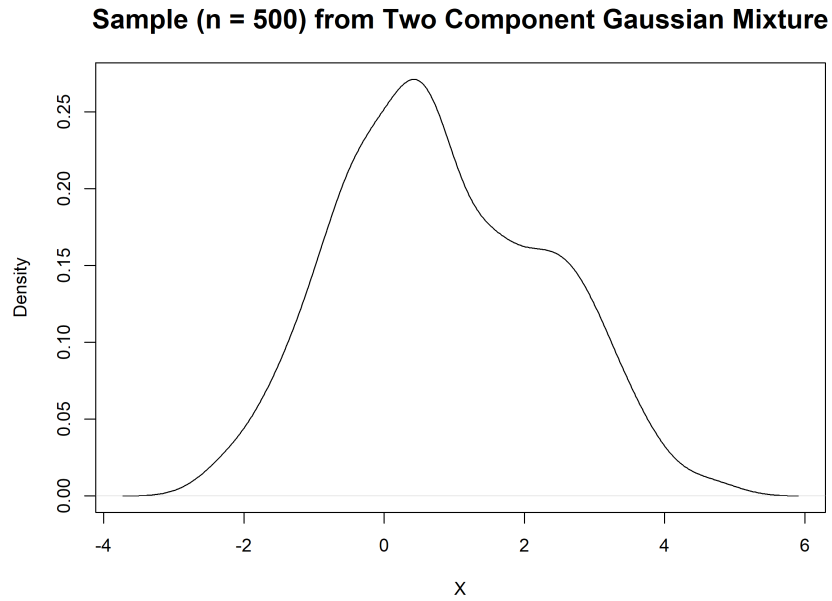


Figure 1.9: Density curve for a sample of $n = 500$ points from a two-component Gaussian mixture with $\pi = 0.7$, $\mu_1 = 0$, $\mu_2 = 2.5$, and $\sigma_1 = \sigma_2 = 1$.

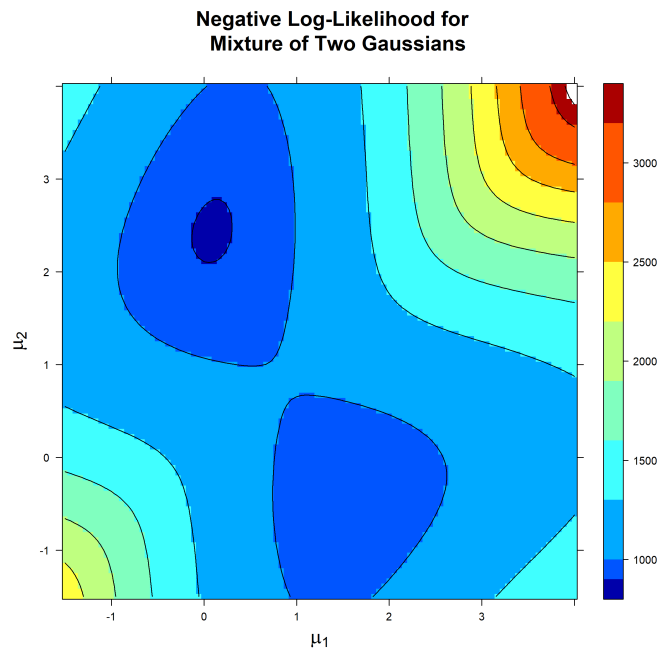


Figure 1.10: Negative log-likelihood for a sample of $n = 500$ points from a two-component Gaussian mixture with $\pi = 0.7$, $\mu_1 = 0$, $\mu_2 = 2.5$, and $\sigma_1 = \sigma_2 = 1$.

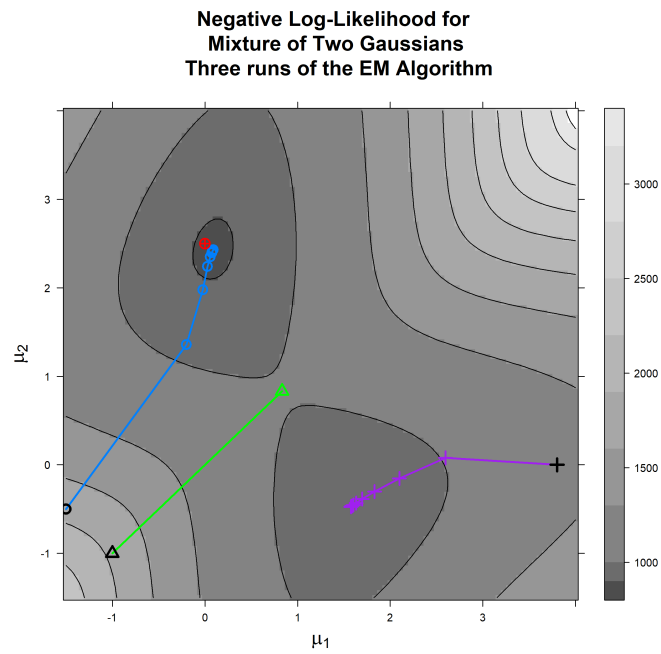


Figure 1.11: Three runs of the EM algorithm on the log-likelihood for the EM algorithm. The red point is the true parameter value. The blue EM chain reaches the global mode. The green chain terminates at a saddle point. Purple chain reaches a local mode.

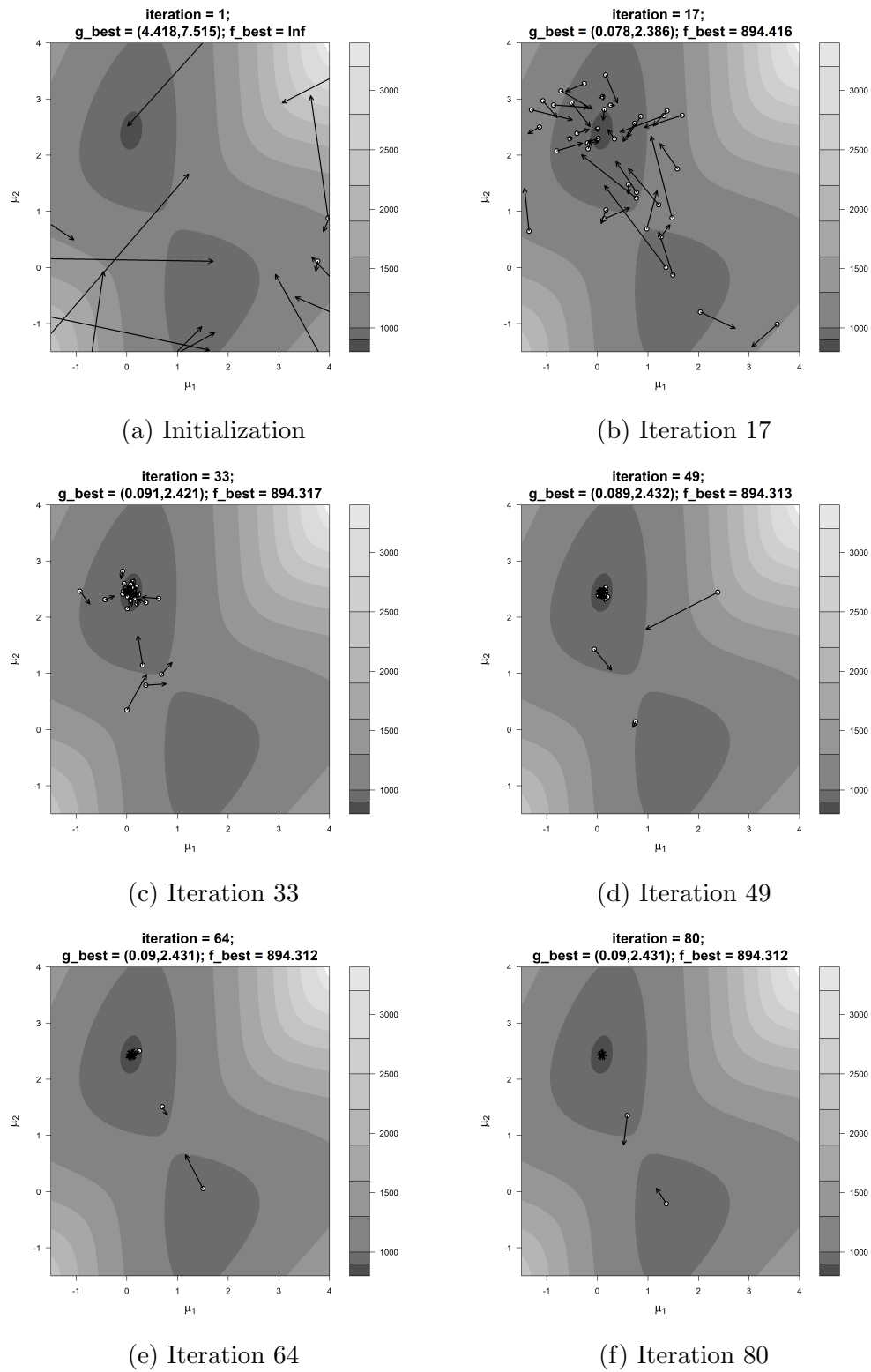


Figure 1.12: Application of SPSO 2011 to minimize the negative log-likelihood for the two-component Gaussian mixture. Particles were initialized in the range $[-10, 10]$ in both dimensions.

CHAPTER TWO

PRIMER ON OPTIMAL DESIGN OF EXPERIMENTS

2.1 Introduction

Jensen (2018) [51] provides a recent paper in Quality Engineering titled “Open problems and issues in optimal design”. The paper is aimed at practitioners and theoretical statisticians. The author describes that in recent years, in part due to advances in computing technology, the optimal design paradigm has achieved maturity and is gaining in popularity over the classical design paradigm. Similar to Goos and Jones (2011) [44], Jensen highlights that the attractiveness of the optimal design paradigm is its flexibility in working the practical constraints of the design problem (e.g., cost, time) into the design formulation problem. Jensen provides a nice summary of the state-of-technology with respect to optimal design and also highlights why some practitioners may be hesitant to adopt this approach. The paper concludes with a list of open problems in optimal design, the solutions of which will be beneficial to increasing the use of optimal design among practitioners.

The purpose of this chapter is to describe the main topics, notation, metrics, and provide a brief summary of common approaches to optimal design. We provide summaries of the following topics:

1. brief discussion of classical vs. optimal design,
2. useful notation that will be used throughout the remainder of this dissertation,
3. the distinction between continuous/approximate designs vs. exact designs,
4. the general equivalence theorem and classical design-generation algorithms,

5. commonly used optimality criteria for constructing designs for linear models with our focus being second-order response surface designs, and
6. a brief high-level summary of common algorithms used to construct optimal designs.

Our notation follows closely to that in a modern text on the subject by Atkinson, Donev, and Tobias (2007) [6].

2.2 Classical vs. Optimal Designs

Classical designs comprise a large catalog which a reader would encounter in common texts on experimental design and response surface methodology (RSM). These designs include (among others): central composite designs (CCD), Box-Behnken designs, Hoke designs, and the set of full and fractional factorial designs. They are often motivated by attempting to ‘cover’ a design space via simple geometries, often a hypercube or hypersphere. Factorial designs represent a general framework for construction of experimental designs used to fit models at various stages of response surface methodology. Let K represent the number of factors, and say that each factor will be studied at 2 levels. This constructs the family of 2^K full factorial designs which, geometrically, contain points at the vertices of a K dimensional hypercube. 2^K designs represent a strong framework for which to fit first order models with interactions, a model widely employed in screening studies, but they do not allow for fitting higher order polynomial terms, such as quadratic terms, which are required for estimating the optima of a response surface. To accomplish this one may implement a 3^K factorial. However, a 3^K design is expensive as it requires $N = 3^3 = 27$ experimental runs for $K = 3$ factors. To address the design size/cost issue, fractional factorial designs are constructed from a fraction of a full factorial design,

thereby reducing the experimental cost. These designs must be generated carefully, as they often require or invoke assumptions such as all higher order terms of a specified order are negligible. They make these assumptions because fractional factorials will not allow for estimating the specified higher order terms, and, in fact, these terms are aliased with one or more model terms [68]. The central composite design (CCD) is a go-to design for fitting second order response surface models. The motivation behind construction of a CCD was to enable fitting the second order model while reducing the experimental cost (i.e., the number of experimental runs N) as compared to the number required by a 3^K full factorial design. CCDs are constructed by first taking a set of runs from a 2^K factorial or fractional factorial design, and then augmenting with center and axial points. Depending on the location of the axial points, the geometry of this design may follow a hypercube (i.e., face centered CCD) or a hypersphere (i.e., the axial points are placed at the same or similar distance from the centroid as the vertices of the cube defined by the factorial points). The Box-Behnken design was developed under a hypersphere geometry with the motivation to enable fitting of the second-order model with reduced N (while also being less costly than the CCD). In short: all of the classical designs briefly discussed here focus on K factors and then placing N runs in the design space while noting what order of model may be fit from the selected N points.

Optimal design (sometimes called algorithmic design, or computer aided design) poses the design problem from a more comprehensive framework [6, 13, 44, 68, 78]. This framework has two main components:

1. specification of the model which will be fit to the data (e.g., first-order, first-order plus interaction, or second-order model). This includes specification of the number of experimental factors K , and

2. specification of an *optimality criterion*. This should be congruent with the needs of the design. For example, if we are conducting a factor screening study via fitting a first-order model with interactions, we desire the design which minimizes some function of the standard errors of the model coefficient estimates, thereby increasing the power of the study to detect the important experimental factors in a minimum number of runs. Further, as in the optimization phase of RSM, one uses the model to predict the response surface at points interior to the design region but which were not sampled in the design. Therefore, in this setting, one desires a design that will minimize some function of the prediction variance.

Another aspect of optimal design, which differs between the continuous and exact design frameworks, is the specification of the number of design points. In the continuous design framework, one specifies M (which represents the number of *unique* design points), or views this as an additional parameter in the optimization. In contrast, in the exact design framework, one specifies (fixes) N which represents the total number of design points (not necessarily unique) that can be afforded in the experimental campaign [6, 13, 78].

With those components specified, optimal designs are computed via some algorithm used to optimize an objective called an *optimality criterion*. The precise way of computing optimal designs is dependent upon which modeling framework is being employed, specifically *continuous* designs or *exact* designs. We will expand on the difference between continuous and exact designs in subsequent sections, and will describe a number of optimality criteria commonly used to generate optimal designs in practice.

2.3 Useful Notation

Let $\mathcal{X}$ denote a design space. It is common to assume that the design factors have been scaled to $[-1, 1]$ in which case the design space is the K -dimensional hypercube $\mathcal{X} = [-1, 1]^K$ for K factors. Let $\mathbf{x}' \in \mathcal{X}$ denote a $1 \times K$ point in the design space. For N specified design points, the set of distinct design points $\mathbf{x}'_i$, $i = 1, 2, \dots, N$ are organized as the rows of the $N \times K$ design matrix

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ \vdots \\ \mathbf{x}'_N \end{pmatrix}.$$

Next, we provide some notation that clarifies the difference between the design space and the model space—the latter of which is required to evaluate the properties and the quality of a candidate design $\mathbf{X}$. Let p represent the number of parameters of the model that one intends to fit to the data. We denote the $N \times p$ model matrix as $\mathbf{F}(\mathbf{X})$. Here we have emphasized that the model matrix is a function of the design matrix. For simplicity, however, from this point forward we will simply represent the model matrix by the shorthand $\mathbf{F}$. A similar device is required for a model vector, which is a function that maps a candidate design point to the model space. We adopt the notation $\mathbf{f}'(\mathbf{x}')$ which is a $1 \times p$ row-vector.

We provide an example for the case of an N point design with $K = 3$ design factors on the second order model to clarify the use of this notation. In this case there are

$$p = \binom{K+2}{2} = \binom{5}{2} = 10$$

parameters in the second-order model [13, 68]. The design matrix for this model is

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ \vdots \\ \mathbf{x}'_N \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ & \vdots & \\ x_{N1} & x_{N2} & x_{N3} \end{pmatrix}.$$

The second-order model, for a single design point $\mathbf{x}' = (x_1 \ x_2 \ x_3)$ in scalar form is written

$$y = \beta_0 + \sum_{i=1}^K \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \beta_{ij} x_i x_j + \sum_{i=1}^K \beta_{ii} x_i^2 + \epsilon,$$

and so the mapping of the design point $\mathbf{x}'$ into the model space is the $1 \times p$ vector

$$\mathbf{f}'(\mathbf{x}') = \begin{pmatrix} 1 & x_1 & x_2 & x_3 & x_1 x_2 & x_1 x_3 & x_2 x_3 & x_1^2 & x_2^2 & x_3^2 \end{pmatrix}.$$

The mapping of the design matrix into the model matrix is

$$\mathbf{F}(\mathbf{X}) = \mathbf{F} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{13} & x_{11}x_{12} & x_{11}x_{13} & x_{12}x_{13} & x_{11}^2 & x_{12}^2 & x_{13}^2 \\ 1 & x_{21} & x_{22} & x_{23} & x_{21}x_{22} & x_{21}x_{23} & x_{22}x_{23} & x_{21}^2 & x_{22}^2 & x_{23}^2 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots & \\ 1 & x_{N1} & x_{N2} & x_{N3} & x_{N1}x_{N2} & x_{N1}x_{N3} & x_{N2}x_{N3} & x_{N1}^2 & x_{N2}^2 & x_{N3}^2 \end{pmatrix}.$$

This enables writing the full second-order linear model in the N design points in matrix form

$$\mathbf{y} = \mathbf{F}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\boldsymbol{\beta}$ is a $p \times 1$ vector of parameters. Under the ordinary least squares (OLS) framework, it is assumed that $\boldsymbol{\epsilon} \sim \mathcal{N}_N(\mathbf{0}, \sigma^2 \mathbf{I}_N)$ where $\mathcal{N}_N$ denotes the N -dimensional multivariate normal distribution. The least-squares estimate of the $\boldsymbol{\beta}$'s (which in this

framework are also ML estimates) is

$$\hat{\boldsymbol{\beta}} = (\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}'\mathbf{y},$$

which have variance-covariance

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{F}'\mathbf{F})^{-1}.$$

A prediction of the response at new point $\mathbf{x}'$ is denoted as $\hat{y}(\mathbf{x}')$, and the variance of the mean predicted value is

$$\text{Var}(\hat{y}(\mathbf{x}')) = \sigma^2 \mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}').$$

The total information matrix for $\boldsymbol{\beta}$, specifically $\mathbf{F}'\mathbf{F}$, plays an important role in optimal design of experiments. We will expand on this role in the subsequent sections.

2.4 Continuous vs. Exact Designs

In this section we contrast the notation and modeling frameworks of continuous and exact designs.

2.4.1 Continuous Designs

Continuous designs are represented by a probability measure $\xi \in \Xi$ over $\mathcal{X}$ where Ξ represents the space of continuous designs. For a design that contains M distinct design points the measure ξ is written

$$\xi = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_M \\ w_1 & w_2 & \cdots & w_M \end{pmatrix}$$

where

$$\int_{\mathcal{X}} \xi(\mathbf{x}) d\mathbf{x} = 1, \quad 0 \leq w_i \leq 1 \quad \forall i, \quad \sum_{i=1}^M w_i = 1.$$

Optimal designs derived from a continuous design framework are often viewed as asymptotic optimal designs. That is, if the number of design points $N \rightarrow \infty$ then optimal designs derived under this framework represent the theoretical optima. If the practitioner chooses to implement a continuous design then the number of replicates of design point i they should ideally implement is the fraction $N_i = \lceil Mw_i \rceil$. However, the practitioner must round the number of replicates to an integer, thereby making the properties of the actually implemented design suboptimal (perhaps slightly if M is large) from the properties of the optimal continuous design. The average information conveyed by model ξ is the $p \times p$ matrix

$$\begin{aligned} \mathbf{M}(\xi) &= \int_{\mathcal{X}} \mathbf{f}(\mathbf{x}') \mathbf{f}'(\mathbf{x}') d\mathbf{x} \\ &= \sum_{i=1}^M w_i \mathbf{f}(\mathbf{x}') \mathbf{f}'(\mathbf{x}'). \end{aligned} \tag{2.1}$$

2.4.2 Exact Designs

In practice all designs are exact. A fixed integer N design points are run in the experiment [13, 78]. A notation commonly used for this, in terms of measure [6], is

$$\xi_N = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ \frac{r_1}{N} & \frac{r_2}{N} & \cdots & \frac{r_n}{N} \end{pmatrix}$$

where r_i represents the integer number of trials of experiment i run at design point $\mathbf{x}'_i$ and $\sum_{i=1}^M r_i = N$. Exact optimal designs may be approximately determined using this measure framework. If ξ^* represents a continuous optimal design, then, adopting this design in practice will yield an acceptable exact optimal design when using $r_i = \lceil Nw_i \rceil$

as the number of replications of design point i (in which case we have $\frac{r_i}{N} \approx w_i$) provided that N is moderate to large. In general the design weights w_i are not rational and so it is not possible to find optimal exact designs that match (and thus inherit) all properties of the continuous optimal design ξ^* [6].

Thus, it is appealing to formulate exact optimal designs separate from the continuous design framework. Instead of using the measure notation ξ_N to represent exact designs, we will refer directly to the $N \times K$ design matrix $\mathbf{X}$. The notation for the search space of all exact designs is as follows. While a design point $\mathbf{x}' \in \mathcal{X} = \times_{k=1}^K [-1, 1] = [-1, 1]^K$, the space of candidate design matrices is the cross-product of K -cubes, specifically:

$$\mathbf{X} \in \times_{j=1}^N \mathcal{X} = \times_{j=1}^N [-1, 1]^K = [-1, 1]^{NK} = \mathcal{X}^N. \quad (2.2)$$

Eq. 2.2 makes clear that finding an exact optimal design is an NK dimensional search. The average information conveyed by the exact design $\xi_N = \mathbf{X}$ is

$$\mathbf{M}(\mathbf{X}) = \frac{\mathbf{F}'\mathbf{F}}{N}. \quad (2.3)$$

2.5 The General Equivalence Theorem

The General Equivalence theorem is a foundational result in the theory of optimal design for continuous models [6, 58]. We here summarize the theorem as presented in Atkinson, Donev, and Tobias (2007) [6].

Let $\mathcal{X}$ be a compact space. Consider a compact and differentiable general measure of imprecision $\Psi(\xi)$. Let the measure $\bar{\xi}$ put unit mass at the point $\mathbf{x}'$

and let the measure ξ' be

$$\xi' = (1 - \alpha)\xi + \alpha\bar{\xi}, \quad \alpha \in (0, 1].$$

The average information of ξ' is given by

$$\mathbf{M}(\xi') = (1 - \alpha)\mathbf{M}(\xi) + \alpha\mathbf{M}(\bar{\xi}).$$

The derivative of Ψ in the direction of $\bar{\xi}$ is

$$\phi(\mathbf{x}', \xi) = \lim_{\alpha \rightarrow 0^+} \frac{1}{\alpha} \left\{ \Psi[(1 - \alpha)\mathbf{M}(\xi) + \alpha\mathbf{M}(\bar{\xi})] - \Psi[\mathbf{M}(\xi)] \right\}.$$

Theorem 1. The General Equivalence Theorem (GET) *Under the above stated conditions, the following statements about ξ^* are equivalent:*

1. *The design ξ^* minimizes $\Psi(\mathbf{M}(\xi))$.*
2. *The design ξ^* maximizes the minimum of $\phi(\mathbf{x}', \xi)$ over $\mathcal{X}$.*
3. *The minimum of $\phi(\mathbf{x}', \xi^*)$ over $\mathcal{X}$ is 0. This minimum occurs at the points of support of the design (i.e., where $w_i \neq 0 \quad \forall i$).*

Item 3. of the GET implies a fourth result:

4. *For any non-optimal design ξ , the minimum over $\mathcal{X}$ of $\phi(\mathbf{x}', \xi) < 0$.*

The GET is used primarily as a tool for checking the properties of continuous designs, often in hopes that a candidate continuous design is an optimal design. A practical result of the theorem that is commonly applied is that optimal continuous designs that minimize the determinant of $\mathbf{M}$ will also minimize the maximum value of the scaled prediction variance. Unfortunately, this result does not hold in general for

exact designs of size N , and so this theorem can only be used as an approximate tool for checking the properties of exact designs (e.g., for large N). In light of this, practitioners, who are often constrained by costs and hence must deal with small to moderate N scenarios, would like a tool to generate and understand the properties of exact designs.

2.6 Some Optimality Criteria

In this section we discuss some design optimality criteria. We will focus on those most commonly used by practitioners. The criteria will be presented as formulae under the exact design framework. For the continuous design analogues, see references [6, 88]. Note that, in this section, the notation $\mathbf{M}(\mathbf{X})$ refers directly to the expression of the design information matrix for an arbitrary exact design $\mathbf{X}$ as given in Eq. 2.3. Further, the notation $\mathbf{X}^*$ will be used to denote the optimal exact design.

2.6.1 D -optimality

The D -score of an arbitrary design $\mathbf{X}$ is defined as the determinant of the inverse of the design information matrix:

$$\begin{aligned} D(\mathbf{X}) &:= |\mathbf{M}^{-1}(\mathbf{X})| \\ &= N^p |(\mathbf{F}'\mathbf{F})^{-1}|. \end{aligned} \tag{2.4}$$

This yields the definition of the D -optimal design $\mathbf{X}^*$

$$\begin{aligned} \mathbf{X}^* &:= \operatorname{argmin}_{\mathbf{X} \in \mathcal{X}^N} D(\mathbf{X}) \\ &= \operatorname{argmin}_{\mathbf{X} \in \mathcal{X}^N} |\mathbf{M}^{-1}(\mathbf{X})| \\ &= \operatorname{argmin}_{\mathbf{X} \in \mathcal{X}^N} N^p |(\mathbf{F}'\mathbf{F})^{-1}|. \end{aligned} \tag{2.5}$$

As a D -optimal design is that design which minimizes the volume of the inverse of the design information matrix (which is proportional to $\text{Var}(\hat{\boldsymbol{\beta}})$), these designs minimize the joint uncertainty in the parameter estimates and so are a popular choice for screening designs. They also are generally popular because of their ease of computation [78].

As the scale of $D(\mathbf{X})$ is arbitrary and depends on the number of design points N and number of model parameters p , it is customary to scale the D -score to the *efficiency* scale

$$D_{\text{eff}}(\mathbf{X}) = 100 \times \left(\frac{D(\mathbf{X}^*)}{D(\mathbf{X})} \right)^{\frac{1}{p}} \in [0, 100]. \quad (2.6)$$

Similarly, the quality of two designs $\mathbf{X}_1$ and $\mathbf{X}_2$ with respect to the D -criterion can be compared relatively as follows

$$D_{\text{rel-eff}}(\mathbf{X}_1, \mathbf{X}_2) = 100 \times \left(\frac{D(\mathbf{X}_2)}{D(\mathbf{X}_1)} \right)^{\frac{1}{p}}. \quad (2.7)$$

2.6.2 G -optimality

A G -optimal design is that design which minimizes the maximum scaled prediction variance over all points of prediction in $\mathcal{X}$. Recall that the prediction variance is

$$\text{Var}(\hat{y}(\mathbf{x}')) = \sigma^2 \mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1} \mathbf{f}(\mathbf{x}')$$

for some point of prediction $\mathbf{x}' \in \mathcal{X}$. The scaled prediction variance (SPV) removes the scale parameter σ and scales to N by multiplying by the factor $N/\hat{\sigma}^2$. SPV is defined as (for a candidate design $\mathbf{X}$)

$$\begin{aligned} \text{SPV}(\mathbf{x}') &:= \mathbf{f}'(\mathbf{x}')\mathbf{M}^{-1}(\mathbf{X})\mathbf{f}(\mathbf{x}') \\ &= N\mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}'). \end{aligned} \quad (2.8)$$

The G -score of an arbitrary design $\mathbf{X}$ is defined as the maximum scaled prediction variance over all points of prediction $\mathbf{x}' \in \mathcal{X}$

$$\begin{aligned} G(\mathbf{X}) &:= \max_{\mathbf{x}' \in \mathcal{X}} \text{SPV}(\mathbf{x}') \\ &= \max_{\mathbf{x}' \in \mathcal{X}} N \mathbf{f}'(\mathbf{x}') (\mathbf{F}' \mathbf{F})^{-1} \mathbf{f}(\mathbf{x}'). \end{aligned} \quad (2.9)$$

Eq. 2.9 indicates that for a fixed candidate design $\mathbf{X}$, scoring the candidate on the G scale is itself an optimization problem.

The G -optimal design $\mathbf{X}^*$ is that design which minimizes, over all designs $\mathbf{X} \in \mathcal{X}^N$, the maximum scaled prediction variance, namely

$$\begin{aligned} \mathbf{X}^* &:= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} G(\mathbf{X}) \\ &= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} \max_{\mathbf{x}' \in \mathcal{X}} \text{SPV}(\mathbf{x}') \\ &= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} \max_{\mathbf{x}' \in \mathcal{X}} N \mathbf{f}'(\mathbf{x}') (\mathbf{F}' \mathbf{F})^{-1} \mathbf{f}(\mathbf{x}'). \end{aligned} \quad (2.10)$$

Eq. 2.10 shows that finding the G -optimal design is a minimax problem. This optimization has proved notoriously difficult to solve because neither of the optimizations required to compute $\mathbf{X}^*$ are convex. We will expand further the difficulties of computing G -optimal designs in Chapter 4.

The scale of G for an arbitrary design has known bounds. Meyers et al. [68] demonstrate that the lower bound on $G(\mathbf{X})$ is

$$G(\mathbf{X}) = \max_{\mathbf{x}' \in \mathcal{X}} \text{SPV}(\mathbf{x}') \geq p; \quad (2.11)$$

that is, the smallest that the maximum scaled prediction variance of candidate design $\mathbf{X}$ may be is p , the number of parameters (although not all G -optimal designs will

achieve this lower bound). Further, if design $\mathbf{X}^*$ is G -optimal, then for this design

$$\text{SPV}(\mathbf{x}') = p$$

at all diagonals of the hat matrix $\mathbf{F}(\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}'$ and

$$\text{SPV}(\mathbf{x}') \leq p$$

at all other points of prediction $\mathbf{x}' \in \mathcal{X}$ [68]. The result stated in 2.11 is an important result—it means that the scale of G -optimality is not arbitrary (at least up to the number of parameters in the model p). It is customary to exploit this fact and work on the G -efficiency scale,

$$G_{\text{eff}}(\mathbf{X}) = 100 \times \frac{p}{G(\mathbf{X})}, \quad (2.12)$$

in order to gauge the quality of a candidate design $\mathbf{X}$. For continuous designs ξ , the GET states that

$$\xi^* = \underset{\xi \in \Xi}{\text{argmin}} G(\xi) = \underset{\xi \in \Xi}{\text{argmin}} D(\xi)$$

with

$$G(\xi^*) = p.$$

That is, for continuous designs, the GET states that a G -optimal design is also D -optimal, and the maximum scaled prediction variance for this model is the number of parameters p .

2.6.3 A -optimality

The A -criterion is similar to the D -criterion in that it seeks to minimize a function of the variance of the $\hat{\boldsymbol{\beta}}$'s. While D is concerned with the size of the scaled variance-covariance of the parameter estimates, A is only concerned with the total

combined uncertainty of the parameter estimates and seeks to minimize the average of this quantity. Thus A is defined as

$$\begin{aligned} A(\mathbf{X}) &:= \text{tr}\{\mathbf{M}^{-1}(\mathbf{X})\} \\ &= N \text{tr}\{(\mathbf{F}'\mathbf{F})^{-1}\} \end{aligned} \quad (2.13)$$

where $\text{tr}\{\cdot\}$ is the matrix trace operator. While easier to compute than G , the use of A as a design criterion is less popular in practice than D . The A -optimal design $\mathbf{X}^*$ is defined as

$$\begin{aligned} \mathbf{X}^* &:= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} A(\mathbf{X}) \\ &= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} N \text{tr}\{(\mathbf{F}'\mathbf{F})^{-1}\}. \end{aligned} \quad (2.14)$$

It is customary to gauge the quality of a candidate design $\mathbf{X}$ on the A -optimality scale by comparing it to a hypothetical orthogonal design (which has trace p in the information matrix) [13, 88]

$$A_{\text{eff}}(\mathbf{X}) = 100 \times \frac{p}{A(\mathbf{X})}. \quad (2.15)$$

2.6.4 IV -optimality

IV stands for ‘integrated variance’ and so the IV -criterion focuses on designs that minimize the average prediction variance over $\mathcal{X}$. This criterion seems to have gained in popularity in recent years among practitioners due to its focus on minimizing a function of the prediction variances (important in response surface modeling) and the fact that it is relatively easier to generate IV -optimal designs as compared to

G -optimal designs [51, 78]. The IV -criterion for candidate design $\mathbf{X}$ is defined as

$$\begin{aligned}
 IV(\mathbf{X}) &:= \frac{1}{V} \int_{\mathcal{X}} \text{SPV}(\mathbf{x}') d\mathbf{x}' \\
 &= \frac{N}{V} \int_{\mathcal{X}} \mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}') d\mathbf{x}' \\
 &= \frac{N}{V} \int_{\mathcal{X}} \text{tr} \left\{ \mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}') \right\} d\mathbf{x}' \\
 &= \frac{N}{V} \int_{\mathcal{X}} \text{tr} \left\{ \mathbf{f}(\mathbf{x}')\mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1} \right\} d\mathbf{x}' \\
 &= \frac{N}{V} \text{tr} \left\{ (\mathbf{F}'\mathbf{F})^{-1} \int_{\mathcal{X}} \mathbf{f}(\mathbf{x}')\mathbf{f}'(\mathbf{x}') d\mathbf{x}' \right\}
 \end{aligned} \tag{2.16}$$

where $V = \int_{\mathcal{X}} d\mathbf{x}'$ yields the volume of the design space $\mathcal{X}$. The IV -optimal design $\mathbf{X}^*$ is defined as

$$\begin{aligned}
 \mathbf{X}^* &:= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} IV(\mathbf{X}) \\
 &= \underset{\mathbf{X} \in \mathcal{X}^N}{\text{argmin}} \frac{N}{V} \text{tr} \left\{ (\mathbf{F}'\mathbf{F})^{-1} \int_{\mathcal{X}} \mathbf{f}(\mathbf{x}')\mathbf{f}'(\mathbf{x}') d\mathbf{x}' \right\}.
 \end{aligned} \tag{2.17}$$

The quality of a candidate design $\mathbf{X}$ can be measured via its efficiency

$$I_{\text{eff}}(\mathbf{X}) = 100 \times \left(\frac{I(\mathbf{X}^*)}{I(\mathbf{X})} \right). \tag{2.18}$$

Without knowledge of the IV -optimal design $\mathbf{X}^*$ it appears that the IV scale remains arbitrary to specific design problems (i.e., fixed N , k , and p).

2.7 Brief Summary of Common Algorithms use to Generate Exact Optimal Designs

Research into algorithms to generate optimal designs has been ongoing since the subject was introduced by Kiefer (1959) [58, 78].

The General Equivalence Theorem (GET) has been heavily leveraged in work

that focuses on generating continuous optimal designs [6]. The GET yields analytical optimization (via calculus) in some scenarios, specifically, in small continuous optimal designs. Mathematical tractability is quickly lost, however, as M (number of unique design points) and K (dimension of design space) increase. The GET elucidates opportunities to apply gradient descent algorithms in the optimal design search. Such algorithms have been implemented in commercial software such as the OPTTEX procedure in SAS [6]. It's noted that convergence in first order algorithms can be slow. Further, a grid search is often applied in order to find better candidates than an arbitrary starting point. As aforementioned, however, all implemented designs are exact, and so a significant portion of the literature focuses on producing algorithms for generating exact optimal designs.

Algorithms for generating exact optimal designs allow the user to fix N (the number of experiments that can be afforded in an experimental campaign). Simple low dimensional exact optimal design problems can sometimes be approached analytically [6], but mathematical tractability is rapidly lost as N , K , or the model complexity increase [6]. Further, the optimality criteria shown in the previous section are not only high dimensional objectives for large N , K , but they are also multimodal, meaning that gradient descent searches will perform poorly by getting trapped at local optima and may not find the global optimizer $\mathbf{X}^*$. Computing time is also highlighted as a problem in exact optimal design searches as the search is an NK dimensional optimization problem [6]. To mitigate the computing time, it has become common practice to approximate continuous region $\mathcal{X}$ via a set of discrete candidate points [6]. The grid is often coarse over the design region and often includes points from a factorial design framework (such as points from a 3^K factorial design). Algorithms which require the user to specify a candidate set to optimize over are collectively referred to as *point exchange algorithms* [6]. Generically, in the search for an N

point optimal exact design, the user specifies a set of N_C candidate points. The initial design is constructed sequentially from a starting design of size N_0 , and then addition or deletion of additional candidate points to arrive at specified design size N . Once the design size is reached, then points in the fixed design are exchanged, and the objective (optimality criterion) is evaluated. Points that improve the optimality criterion are retained in the candidate design. The most straightforward exchange algorithm which uses forward and backwards searches (i.e., starting small and ending at N , or starting large and ending at N) are generally not guaranteed to converge to the optimal design (conditioned on the set of candidate points) [6]. To improve upon the selected design, non-sequential algorithms such as DETMAX, Federov, KL and BLKL algorithms, were developed in the 1970s and 80s. Such algorithms are available in many software packages including R and SAS [6, 93]. Due to the use of a candidate set which discretizes $\mathcal{X}$, this approach is said to generate *approximate* optimal designs. The run time of point exchange algorithms increases linearly with the number of candidate points, but exponentially in the number of factors K and so this approach is only feasible for small K .

A drawback of point exchange algorithms is that they are sensitive to the choice of candidate set which discretizes $\mathcal{X}$. Provided that a good candidate set is provided for the given optimality criteria, the generated approximate optimal design will be close to the true exact optimal design. This may be difficult to ensure for criterion more complicated than D -optimality and especially for higher order response surface models [6]. Recognizing this, Borkowski (2003) [13] produced a highly cited paper which opened up a new vein of research in exact optimal design. The author researched the efficacy of evolutionary algorithms for optimization, specifically genetic algorithms (GAs), as an engine for searching for exact optimal designs. In this paper the author took the following perspectives:

1. The space of candidate designs, $\mathcal{X}^N$ can/should be treated as continuous.
2. The author considered small exact second-order response surface models.
3. The author covered $K = 1, 2$, and 3 design factors over a range of N for each scenario.
4. The dimension of the optimization problem gets moderately large quickly, with the largest design scenario considered was $K = 3$ and $N = 16$ yielding a $NK = 48$ dimensional optimization problem.
5. The objectives, specifically the D , G , A , and IV optimality criteria, are high dimensional multimodal objectives.
6. Genetic algorithms have been demonstrated to be effective in finding global optima in such settings.

The author published a catalog of optimal designs for small exact response surfaces, many of which were to that point unknown. The results of this paper have become, and still remain, foundational work and a standard for comparison of new algorithms and new small exact response surface designs. The results published in this paper act as the foundation against which the work of this dissertation thesis will be compared.

The contemporary state-of-the art for generating optimal exact designs (which is implemented in many proprietary statistical software) are *coordinate exchange* algorithms. Myers and Nachtsheim (1995) published “The coordinate-exchange algorithm for constructing exact optimal experimental designs” which presents a (at that time) new approach to generating exact optimal designs [66]. Myers and Nachtsheim were motivated by the following two drawbacks of point-exchange algorithms: 1.) discretizing of $\mathcal{X}$ via a candidate set, and 2.) the computational time

being exponential in K . Goos and Jones (2011) [44] give a heuristic description of the basic coordinate exchange algorithm which we repeat here:

1. Randomly draw a candidate design matrix $\mathbf{X} \sim \mathcal{X}^N$. If the candidate yields a singular design information matrix, redraw until a candidate which yields a non-singular design information matrix is realized.
2. Improve the candidate $\mathbf{X}$, which has $N \times K$ coordinates, by exchanging each with the extreme values of $\mathcal{X}$. Retain the value at that coordinate if it improves the optimality criterion.
3. Repeat the process by looping through all $N \times K$ coordinates until no further improvement in the criterion is observed.
4. As the above steps only result in locally optimal designs, start at step 1. again and repeat many times to improve the chances of finding the globally optimal design.

Goos and Jones (2011) recommend, except for the most simple $K = 1, 2$ design problems, to repeat the coordinate exchange at least 1000 times. Due to the general widespread acceptance of coordinate exchange algorithms by the statistics community, they are the gold standard against which to judge the efficiency and efficacy of new proposed optimal exact design generation algorithms.

2.8 Conclusions

In this chapter we provided a brief high level summary of optimal design of experiments. We provided a contrast between continuous and exact design frameworks. We explicated our notation for the linear model framework and contrasted the difference between the design matrix (which is the subject of our

optimization problems) and the model matrix (which is required to gauge the quality of a candidate design) via this notation. We discussed a number of design criteria (optimality objectives) which are commonly used by practitioners and in contemporary software packages for generating optimal designs. Last, we concluded with a brief heuristic summary of algorithms which are commonly used to search for optimal exact designs.

CHAPTER THREE

EXTENDING PARTICLE SWARM OPTIMIZATION TO GENERATE EXACT
OPTIMAL EXPERIMENTAL DESIGNSABSTRACT

In this chapter we extend PSO to enable optimizing functions with matrix inputs. We provide the mathematical extension in the form of the velocity update equations. This enables PSO as a mechanism for generating exact optimal experimental designs on the hypercube. We provide this solution in the form of new software written in the `Julia` language. We validate the software against the results provided in Borkowski (2003) in a large scale study using the D and A -criteria. The resulting database represents, to the author's knowledge, the first thorough application of 3 standard versions of PSO for generating optimal exact experimental designs. SPSO 2007 is demonstrated to be superior to Basic PSO and SPSO 2011 with respect to success in finding the D -optimal and A -optimal designs for $K = 1, 2, 3$ design factors. The data presented in this chapter constitute a thorough validation of the newly developed `Julia` software.

3.1 Introduction

The objective of the work presented in this chapter is to enable PSO searches for optimal exact experimental designs. We considered using available software, such as the `hydroPSO` package in `R` [96] which is the most comprehensive PSO software publicly available and codifies the over 25 year history of PSO development, giving access to multiple versions of the algorithm (and incidentally is also developed by some of the lead PSO researchers). However, such codes are not tailored directly to

the optimal experimental design problem. Further, while interpreted languages such as R, Matlab, and Python offer rapid development speed, they suffer in computational speed especially when the required operations cannot be vectorized (as an aside we did find `hydroPSO` to be surprisingly fast for moderate dimensions due to some clever coding that took heavy advantage of R's optimized vector operations and application of sync updating). The PSO algorithm relies on two main loops, the `while` loop which iterates the PSO search, and a `for` loop through the particles at each iteration. These loops cannot be broken down into simpler calculations due to the communication dependencies among the particles. In light of these considerations, we were motivated to construct our own software for implementing PSO directly tailored to the optimal design of experiments (DoE) problem. Below we summarize a number of our motivations, decisions, and contributions regarding PSO software development for generating optimal designs:

1. In current PSO formulations, particles are $1 \times K$ vectors and the objective function takes vectors as inputs. In optimal DoE, particles are $N \times K$ matrices and the objective functions take matrices as inputs. The software should accept N and K as input parameters (thus defining the size of design matrix one is searching to optimize). Further, the software should subsume the current PSO approach (optimizing functions with vector inputs). This is accomplished by setting $N = 1$ and K equal to the dimension of the problem.
2. Regarding the programming paradigms, we selected the `Julia` language for software development [8, 9]. `Julia` is a (just in time) compiled language which promises computing speed close to C++, but uses high level programming syntax and hence offers the development speed of R or Python. Our `Julia` code went through three revisions, and in test scenarios is currently nearly 400 times faster

than analogous `R` codes. We anticipate that the user community may further help optimize this software once it is released to the public. Further, `Julia` is built for easy CPU palatalization which will help with running multiple PSO searches quickly.

3. We give the user access to three standard PSO variants as described in Chapter 1: Basic PSO, SPSO 2007, and SPSO 2011. Each version uses synchronous updating [77,95] and applies the published/recommended velocity weight factors ω , c_1 , and c_2 (though these may be specified by the user) [28].
4. Swarm size S , number of iterations, and maximum number of stagnations of the global best solution may be specified by the user. The current stopping criterion is max iterations or max stagnations (whichever occurs first), or a user specified tolerance on a non-zero move of $f(\mathbf{G}_{\text{best}})$.
5. In this chapter we list all `Julia` functions developed during this exercise. We describe the inputs, usage, and output of the main function `swarmSearch_Hypercube` which the user calls to implement a PSO search. All other functions inherit arguments from this function (and so the user does not call them) and, therefore, we only describe the purpose and operation of each utility function. We intend this chapter to be the genesis of help file documentation for future published software.
6. To validate our software, we implemented a large scale study on all design scenarios presented in Borkowski (2003) [13]. We applied two optimality criteria: D and A optimality. We varied the PSO algorithm: Basic PSO, SPSO 2007, SPSO 2011. We also varied the swarm size: $S = 50, 150, 500$ particles. For each design scenario we ran $n_{\text{run}} = 140$ independent PSO searches. Thus, in total, the validation exercise comprised 52,920 independent runs of PSO.

7. We identified the need in the literature for a thorough benchmarking of standard PSO algorithm performance on optimal exact DoE problems. We take the data published by Borkowski (2003) for that purpose and provide a comprehensive performance analysis [13]. We believe this study is a contribution to the literature for two reasons: 1.) Applications of PSO to optimal DoE are relatively new and somewhat limited, focusing mostly on generating approximate/continuous designs and often are proof-of-concept papers which simply illustrate the approach can generate published designs. Therefore, no comprehensive benchmarking of the algorithms to optimal DoE problems has been published; and 2.) All approaches for PSO we are currently aware of in the literature use Basic PSO (i.e., the global communication topology). However, the PSO literature currently considers some version of the local communication topology (e.g., SPSO 2007) as state-of-the-art. The data resulting from this study will be the basis for comparing improvements to the algorithms which we will present in subsequent chapters.

Last we dub the software PSO-OptDoE.

3.2 Extending the PSO Update Equations to Functions with Matrix Inputs

Recall the PSO velocity and position update equations which we restate here as a function of iteration index t :

$$\mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) \tag{inertia} \quad (3.1)$$

$$+ c_1 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{p}_{\text{best},i} - \mathbf{x}_i(t)) \tag{cognitive} \quad (3.2)$$

$$+ c_2 U_K(\mathbf{0}, \mathbf{1}) \odot (\mathbf{g}_{\text{best}} - \mathbf{x}_i(t)) \tag{social} \quad (3.3)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (3.4)$$

where $\odot$ represents the Hadamard product (elementwise multiplication). The quantities $\mathbf{v}$, $\mathbf{x}$, U_K , $\mathbf{p}_{\text{best}}$, and $\mathbf{p}_{\text{best}}$ are all $1 \times K$ vectors. Further, recall the box constraint vectors $\mathbf{l}_b : 1 \times K$ and $\mathbf{u}_b : 1 \times K$ which define the boundaries of the search space in each of the K dimensions.

The objective of this section is to change particles from a $1 \times K$ vector $\mathbf{x}$ to an $N \times K$ design matrix $\mathbf{X}$. We considered using the vectorization operator to map particle matrices into vectors as

$$\mathbf{x}^* = \text{vec}(\mathbf{X}), \quad (3.5)$$

as this would give the optimal design generation problem direct access to available PSO codes such as the domain standard `hydroPSO` in R [96]. A set of functions would be required on the front end (i.e., before the call to PSO) to organize $\mathbf{X}$, to define the constraints $\mathbf{l}_b$ and $\mathbf{u}_b$, and also on the back end to unpack the vec'd solution matrix back into an $N \times K$ matrix.

The `Julia` language which discourages the use of vectorized operations, and in fact encourages the use of `for` loops, and so we adopted a different but equivalent approach. We allow a particle to be an $\mathbf{X} : N \times K$ matrix and organize the S particles in the swarm into an $N \times K \times S$ three-dimensional array. The corresponding velocities and all personal, local, and global best positions are organized similarly. Further, constraints $\mathbf{l}_b$ and $\mathbf{u}_b$ can remain as K vectors with this approach whereby we take the k th element of these vectors to constrain the lower and upper bounds on all N elements of the k th column of $\mathbf{X}$. The update equations under this formulation at

time $t + 1$ for particle i can be written

$$\mathbf{V}_i(t + 1) = \omega \mathbf{V}_i(t) \quad (\text{inertia}) \quad (3.6)$$

$$+ c_1 \mathbf{U} \odot (\mathbf{P}_{\text{best},i} - \mathbf{X}_i(t)) \quad (\text{cognitive}) \quad (3.7)$$

$$+ c_2 \mathbf{U} \odot (\mathbf{G}_{\text{best}} - \mathbf{X}_i(t)) \quad (\text{social}) \quad (3.8)$$

$$\mathbf{X}_i(t + 1) = \mathbf{X}_i(t) + \mathbf{V}_i(t + 1) \quad (3.9)$$

where the $N \times K$ matrix $\mathbf{U} = \{u_{nk}\}$ with $u_{nk} \sim U(0, 1)$ (i.e., all elements are randomly drawn from a uniform distribution). It is evident that equations (3.6-3.9) reduce to Eq.s (3.1-3.4) when $N = 1$ and the resulting vectors are transposed.

In this way we have generalized the PSO algorithm to optimizing functions which take matrices as inputs.

3.3 PSO-OptDoE: List of Julia Functions

In this section we provide a summary of all Julia functions we developed to enable PSO as an engine for searching for exact DoEs on the hypercube. The software is built around the formulation presented in Eq.s (3.6-3.9).

3.3.1 swarmSearch_Hypercube

Description

`swarmSearch_Hypercube` is used to implement PSO searches on a bounded hyper-rectangle. NOTE: this function only performs minimization searches. It can be used on functions that take vector or matrix inputs. Currently, three versions of standard PSO are implemented: Basic, SPSO 2007, SPSO 2011.

Usage

```

swarmSearch_Hypercube(; N, K, S, objective, l_vec, u_vec, max_iter = 5000, w =
1/(2*log(2)), c1 = 0.5 + log(2), c2 = 0.5 + log(2), PSOversion = "Basic", nn
= 3, maxStag = 250, init_method = "random", ngen = 1, printProg = true, output
= "small", order = NaN)

```

Arguments

N: number of design points.

K: dimension of search space.

S: number of particles to run in the swarm search.

objective: a user specified objective function.

l_vec: a K vector containing lower bounds of the irregular hypercube.

u_vec: a K vector containing upper bounds of the irregular hypercube.

max_iter: maximum number of PSO iterations before terminating. Default is 5000.

w, c1, c2: PSO velocity update constants. Currently set at recommended values [28] but may be specified by the user.

PSOversion: string specifying which version of SPSO to use. Permissible values are: `Basic`, `2007`, `2011`.

nn: expected number of neighbor links in the local communication topology. Default is recommended at 3 [28].

maxStag: maximum number of allowed stagnations in global best solution $f(\mathbf{G}_{\text{best}})$ before terminating. Default is 250 but may be user specified.

init_method: string specifying initialization method to place particles in search space at iteration 0. Permissible values are **random** and **RLHCS**. The former will initialize particle positions $\mathbf{X}_i$ completely randomly using a multivariate Uniform distribution as in [28]. The latter will initialize particle positions in a space-filling way (i.e., filling the search space $\mathcal{X}^N$ regularly) based on random Latin hypercube sampling following space-filling optimization via a genetic algorithm.

ngen: if **RLHCS** is specified for initialization, this parameter controls the number of genetic algorithm generations used to optimize the space filling property of the initial set of particles.

printProg: boolean, if **true** then iteration index is printed for the user while the PSO search runs.

output: string indicating how much information to report upon completion. Permissible values are **small** (just solution information) and **all** (full swarm information at last iteration. See Value section below for more explanation.

order: a parameter required for PSO searches for exact optimal designs. This parameter specifies the order of the linear response surface model. A value of 0 specifies the first-order model without interactions, 1 specifies the first-order model with interactions, and 2 specifies the full second-order model. No other values are currently supported.

Value When **output** = **small** n_{iter} , S , $f_{\mathbf{G}_{\text{best}}}$ and $\mathbf{G}_{\text{best}}$ are returned. If **all** then full swarm information at solution is reported including:

1. Array $\left\{ \mathbf{X}_i \right\}_{i=1}^S$ containing all particle positions at the last iteration.

2. Array $\left\{\mathbf{V}_i\right\}_{i=1}^S$ containing all particle velocities at the last iteration.
3. Array $\left\{\mathbf{P}_{\text{best},i}\right\}_{i=1}^S$ containing all particle personal best locations in $\mathcal{X}^N$ at the last iteration.
4. Array $\left\{f(\mathbf{P}_{\text{best},i})\right\}_{i=1}^S$ containing all particle fitness at their personal best locations.
5. Matrix $\mathbf{G}_{\text{best}}$ reporting the global best position found during the search (this is the solution to the optimal DoE problem).
6. Scalar $f(\mathbf{G}_{\text{best}})$ reporting the fitness at the global best position.
7. Array $\left\{\mathbf{L}_{\text{best},i}\right\}_{i=1}^S$ containing all particle local best locations in their communication neighborhoods at the last iteration.
8. Array $\left\{f(\mathbf{L}_{\text{best},i})\right\}_{i=1}^S$ containing fitness at local (neighborhood) best locations at the last iteration.
9. Array $\left\{l_{\text{best.index},i}\right\}_{i=1}^S$ containing the index of the particle with the best fitness (over $1:S$) in each particle's neighborhood set over the entire search history (note that this index need not necessarily match particle indices in the current neighborhood set).
10. Array $\left\{\mathbf{N}_{b,i}\right\}_{i=1}^S$ containing the index set (elements from $1:S$) of the neighbors in the communication neighborhood of particle i . Note that $\mathbf{N}_{b,i}$ can be of different length $\forall i$ because the neighborhood links are randomly drawn.

3.3.2 swarm_initialization

Description

`swarm_initialization` inherits all arguments from `swarmSearch_Hypercube`. This function generates initial particle positions, velocities, communication neighborhoods, and all corresponding fitness measures on `objective`. Currently supported methods for generating starting positions are either a completely random sample from a multivariate Uniform distribution [28] or via random Latin hypercube sampling with high space-filling property.

3.3.3 `genRandDesign_fac`

Description

`genRandDesign_fac` inherits all arguments from `swarmSearch_Hypercube` and is called by `swarm_initialization`. This function produces random initial particle positions based on the Uniform distribution as specified in [28].

3.3.4 `genLHC_fac`

Description

`genLHC_fac` inherits all arguments from `swarmSearch_Hypercube` and is called by `swarm_initialization`. This function produces initial particle positions generated in a space-filling way via random Latin hypercube sampling. The space-filling properties of the initial particle positions in the search space is optimized via a genetic algorithm with the number of generations specified via `ngen`. The approach is given in [92].

3.3.5 `initialVelocity_2007`

Description

`initialVelocity_2007` inherits all arguments from `swarmSearch_Hypercube`. This function generates initial particle velocities for Basic PSO and SPSO 2007 methods according to [28].

3.3.6 initialVelocity_2011

Description

`initialVelocity_2011` inherits all arguments from `swarmSearch_Hypercube`. This function generates initial particle velocities for SPSO 2011 methods according to [28].

3.3.7 update_Vel

Description

`update_Vel` inherits all arguments from `swarmSearch_Hypercube`. For particle i , this function performs the velocity updates (i.e., Eq.s 3.6-3.8) depending on which version of SPSO was specified via the `PSOversion` option [28].

3.3.8 check_velocity

Description

`check_velocity` inherits all arguments from `swarmSearch_Hypercube`. The velocities are limited by a parameter $vmax$ which is currently hardcoded as a K vector giving the length of the hypercube in each dimension [28]. `check_velocity` is called each time particle velocities are updated and returns a boolean value indicating if any particle velocities have exceeded $vmax$.

3.3.9 velLimit

Description

`velLimit` inherits all arguments from `swarmSearch_Hypercube`. The velocities are limited by a parameter $vmax$ which is currently hardcoded as a K vector giving the length of the hypercube in each dimension [28]. `velLimit` is called at particle initialization and each time particle velocities are updated if `check_velocity` returns `true`. If needed this function sets particle i 's velocity $vmax$ in the required dimensions.

3.3.10 check_confinement

Description

`check_confinement` inherits all arguments from `swarmSearch_Hypercube`. This function is called each time particle positions are updated and returns a boolean with `true` indicating that boundaries of the search space were violated by any particle in any dimension [28].

3.3.11 confine_particle

Description

`confine_particle` inherits all arguments from `swarmSearch_Hypercube`. This function is called each time particle positions are updated and `check_confinement` returns true. If particle i has left the search space in any dimension, it is mapped to the boundary of the search space in the respective dimensions [28]. If `PSOversion` is `Basic` or `SPSO 2007`, the absorbing wall is applied. If `PSOversion = 2011` then the reflecting wall is applied [28].

3.3.12 update_pg_bests

Description

`update_pg_bests` inherits all arguments from `swarmSearch_Hypercube`. This function is called for each particle at each iteration of PSO and checks fitness of the current position and updates the personal and global best fitness and corresponding positions if needed. [28].

3.3.13 update_l_best

Description

`update_l_best` inherits all arguments from `swarmSearch_Hypercube`. This function is called for each particle at each iteration of PSO and checks each particle's fitness vs.

that of its local communication neighborhood and updates the corresponding local best fitness and locations if needed [28].

3.3.14 genNeighbors

Description

`genNeighbors` inherits all arguments from `swarmSearch.Hypercube`. This function is called at initialization and at each iteration where stagnation of the global best solution occurs (i.e., at iterations when the particles do not find a better fitness on the objective). This function randomly (re)initializes the local communication neighborhoods of each particle with an expected number of communication links equal to `nn` [28].

3.3.15 randomSphere

Description

`randomSphere` inherits all arguments from `swarmSearch.Hypercube`. This function is called as part of the velocity updates when `PS0version` = 2011. This function generates a random matrix valued point from an $N * K$ multivariate Gaussian distribution with a specified radius [28].

3.3.16 D_criterion

Description

`D_criterion` is an objective function used to score candidate designs during a search for exact D -optimal designs. As this version of PSO performs minimization, this function evaluates the negative of the D -score of a candidate particle $\mathbf{X}$ according to the formula provided in [13].

3.3.17 A_criterion

Description

`A_criterion` is an objective function used to score candidate designs during a search for exact A -optimal designs and may be used in this version of PSO. As this version of PSO performs minimization, this function evaluates the negative of the A -score of a candidate particle $\mathbf{X}$ according to the formula provided in [13].

3.3.18 genModelMat_fac

Description

`genModelMat_fac` is a utility function that maps a design matrix $\mathbf{X} : N \times K$ to a model matrix $\mathbf{F} : N \times p$ of specified order. This function currently supports $K = 1, 2, 3$, and 4 design factors and models of order 0 (first-order no interactions), 1 (first-order plus interactions), and 2 (full second-order response surface model).

3.4 Software Validation and PSO Performance Benchmarking

Initial application of the `Julia` software to standard PSO test functions (with vector inputs, e.g., the Rastrigin function) proved promising and indicated each version of PSO we implemented was able to find the expected solution easily for problems of small to moderate dimension.

In order to provide an extensive validation and benchmarking, we adopt the results reported by Borkowski (2003) “Using a Genetic Algorithm to Generate Small Exact Response Surface Designs” [13]. Recognizing that the exact optimal design problem is mathematically intractable, Borkowski provided the first published exact D , A , G , and IV optimal designs for a number of sample size N scenarios on models with $K = 1, 2$, and 3 factors generated by a genetic algorithm (GA). This paper has become, and remains, something of a standard for comparing new algorithmic

approaches to generating D , A , G , and IV optimal designs. See references [47, 78] for examples. Due to the success of this paper, we view the results as an ideal dataset against which to validate our software and produce benchmark performance measures for Basic PSO, SPSO 2007, and SPSO 2011 for generating small exact optimal designs. The optimal designs published in Borkowski (2003) will be taken as ground truth for the purpose of code validation and PSO performance benchmarking.

In the next section, we present the results of a comprehensive validation study by applying PSO-OptDoE to the D and A -criterion and the design scenarios in Borkowski (2003). These criteria are good starting points for testing the software as D is easy to compute as it only requires computing a determinant to score a design while A represents a step up in complexity as a matrix inversion is also required.

3.4.1 Study Structure

The factors and their levels of each in the validation study are as follows:

1. Optimality Criterion - levels: $\{D, A\}$.
2. PSO Version - levels: $\{\text{Basic PSO, SPSO 2007, SPSO 2011}\}$.
3. Swarm Size - levels: $S \in \{50, 150, 500\}$.
4. Number of Design Factors - $K \in \{1, 2, 3\}$.
5. Number of Design Points - 7 levels per K :

$$K = 1 \implies N \in \{3, 4, 5, 6, 7, 8, 9\}.$$

$$K = 2 \implies N \in \{6, 7, 8, 9, 10, 11, 12\}.$$

$$K = 3 \implies N \in \{10, 11, 12, 13, 14, 15, 16\}.$$

The full scale of the study is the combination of all factor levels above. To clarify the structure of the study, we present the combinations of criterion, PSO version,

and swarm size levels in Table 3.1, and the combinations of number of design factors, number of design points, and number of PSO runs in Table 3.2. For each design scenario we had to specify a number of runs n_{run} , i.e., independent applications, of the PSO search. All tests were run on an Intel Core i7-6700K (four core hyperthreaded) CPU running at 4.0GHz. This setup allows us to occupy seven cores in a parallel computing scenario. In order to reduce computing time, we implemented sets of independent PSO runs in parallel across seven of the cores. The number of PSO runs per core was set at 20, resulting in $n_{\text{run}} = 140$ PSO searches per test scenario. Thus, the size of the validation and performance benchmarking study in total number of PSO searches is

$$\begin{aligned}
 & 2 \text{ criteria} \times 3 \text{ PSO variants} \times 3 \text{ swarm sizes} \times \\
 & 3 \text{ levels of } K \times 7 \text{ levels of } N \times 140 \text{ PSO searches} = \\
 & 52920 \text{ independent PSO searches.}
 \end{aligned}$$

All swarms were initialized via completely random starting points as specified in [28]. The stopping criterion was $\mathbf{G}_{\text{best}}$ stagnation for 100 iterations. Total run time (wall clock) for the full suite of 26460 PSO searches for D -optimal designs was approximately 9 hours 58 minutes. Total run time for the full suite of 26460 PSO searches for A -optimal designs was approximately 13 hours 59 minutes. The quantities $n_{\text{iterations}}$, S , $f(\mathbf{G}_{\text{best}})$, and $\mathbf{G}_{\text{best}}$ were saved in .csv databases and subjected to further analysis.

3.4.2 Software Validation

In this section we present a series of graph panels that illustrate the degree of success of the PSO variants to generate D and A optimal designs for all swarm sizes

and design scenarios considered. The information we present in these graph panels include:

1. distributions of the optimality scores over all PSO runs which are to be compared directly to the GA solutions provided by Borkowski (2003),
2. distributions of the efficiency scores for all PSO runs (the GA solution is taken as the optimal design in the calculation of efficiency of the PSO solutions), and
3. estimates of probability of success of each PSO algorithm where success is defined as finding the GA solution).

3.4.2.1 D -optimal Designs Generated by PSO

In Figure 3.1, we present a set of graph panels which illustrate the PSO search results for D -optimal designs on the $S = 50$ swarm size. The top set of graph panels in this figure show the distributions of the PSO solutions over each run as blue points overlayed on boxplots. Rows of the graph panels correspond to the SPSO version, while columns correspond to K , the number of design factors. The number of design points N is indicated in the horizontal axis of each panel. The vertical scale is the D -criterion as calculated in Borkowski (2003) [13]. Note that using the formulation in that paper, a higher D -score indicates a better design. The values of the D -optimal design as found by the genetic algorithm (GA) are indicated by the large red dot [13]. The results presented in this graphic are encouraging.

The first column of graph panels correspond to $K = 1$ and the seven design point N scenarios. It can be seen that the distributions of the $n_{\text{run}} = 140$ PSO solutions, for each of the three PSO variants, are stacked tightly (i.e., with little to no variation) around the GA solution, indicating any version of PSO has a high success rate for the lower dimensional optimal design problems using the D -criterion. Moving to the

second column for the $K = 2$ factor design scenarios, it can be seen that the PSO solutions for each of the algorithms are now more dispersed than the $K = 1$ solutions, but they still appear close to the GA solution. We note that SPSO 2011 exhibits the most variability. Moving to $K = 3$ design factors, the dispersion of PSO solution increases, and while Basic and SPSO 2007 still sometimes achieve the GA solution, we can see that SPSO 2011 starts to have difficulty in reproducing the GA solution as evidenced by the distribution of solutions shifted downward from the GA D -optimal value.

To better illustrate the degree to which the PSO variants can produce solutions near the global D -optimal designs for these scenarios, we translated the PSO solutions to the D -efficiency scale in the bottom set of panels. In these panels, the GA D -optimal design was taken as the comparator in the D -efficiency calculation, and so a PSO solution that does not achieve the GA solution will have an efficiency $< 100\%$. The results appear to be encouraging for the Basic and SPSO 2007 using only $S = 50$ particles to search for D -optimal designs up to dimension $K * N = 3 * 16 = 48$. It can be seen in the corresponding D -efficiency plots that Basic PSO easily generates designs with 80% D -efficiency or greater and medians around 95% while SPSO 2007 easily generates designs with 85% efficiency or greater and medians at 95% or greater. SPSO 2011 is not as successful, however, as the distribution of D -efficiency for the higher dimension searches appear to have a medians near 85-90%, dropping as low as 70% in some cases, and the D -optimal design is never found by this algorithm for $S = 50$ swarm size searches.

In Figure 3.2 we present a set of graph panels which illustrate the PSO search results for D -optimal designs on the $S = 150$ swarm size case. Generally, as compared to the $S = 50$ cases, the $S = 150$ cases exhibit higher success and efficiency in generating optimal designs. Basic PSO can be seen to easily generate D -optimal

designs with higher than 90% efficiency for many of the higher dimensional cases while SPSO 2007 solutions are highly dense above 90 to 95% efficiency for the same scenarios with the medians approaching 100% efficiency. The ability of SPSO 2011 to produce highly efficient D -optimal designs with $S = 150$ swarm sizes has improved with the minimum efficiency for many of the higher dimensional design scenarios increasing to 80% as compared to the $S = 50$ runs. We note, however, that SPSO 2011 still appears to have trouble finding the D -optimal design as found by the GA for the higher dimensional design scenarios.

We present a set of graph panels which illustrates the PSO search results for D -optimal designs on the $S = 500$ swarm size case in Figure 3.3. Basic PSO seems to have benefited little from the increase in swarm size from $S = 150$ to $S = 500$ for the $K = 3$ (higher dimensional) design scenarios. This is evidenced by very similar patterns in the boxplots for these cases. In contrast, SPSO 2007 seems to achieve some benefit from the larger swarm size given that the boxplots of the D -optimal solutions have become more dense near the GA solution for the $S = 500$ $K = 3$ cases as compared to the $S = 150$ $K = 3$ cases. SPSO 2011 does not seem to derive much benefit from the larger swarm sizes for the $K = 3$ scenarios as the distributions of the D -efficiencies for the $S = 500$ runs appear very similar to those from the $S = 150$ runs.

As we initialized the swarms completely randomly we may view the PSO solutions $f(\mathbf{G}_{\text{best}})$ as a random variable. If we take the GA solutions as ground truth, it is possible to estimate the probability of success of the PSO variants. Define the event success in run r as

$$S_r = \{f(\mathbf{G}_{\text{best}}) \geq f(\mathbf{X}_{\text{GA}})\}$$

where $\mathbf{X}_{\text{GA}}$ is the optimal design found by the GA as reported in Borkowski (2003)

[13]. Then an estimate of the probability of success of the PSO algorithm (conditioned on the design scenario, swarm size, and optimality criterion is

$$\widehat{\Pr}\{\text{success}\} = \frac{\sum_{r=1}^{n_{\text{run}}} I(S_r)}{n_{\text{run}}} \quad (3.10)$$

where $I(\cdot)$ is the indicator function. Binomial theory applies (e.g., for computing standard errors and confidence intervals).

We present estimates of probability of success as a function of the dimension of the optimization problem (i.e., the product $K * N$), PSO algorithm, and swarm size S in Figure 3.4. This summary reveals clear insights regarding the performance of these PSO variants for generating D -optimal designs. First, we note that the estimates of probability of success for SPSO 2011 drop rapidly by the time the dimension of the search hits 10 which is consistent with the observations we made regarding the boxplots of the PSO solutions. Next, we see that, for any swarm size, the success probability of Basic PSO and SPSO 2007 decrease (generally) as a function of dimension. This is not surprising as the generation of D -optimal designs is a complicated high-dimensional optimization of a multimodal objective. Further, the probability of success estimate curves indicate that both Basic PSO and SPSO 2007 appear to receive some benefit in increasing the swarm size—evidenced by probability of success curves for $S = 500$ size swarms being higher than for the other swarm sizes at most dimensions. SPSO 2007 appears to be generally more successful than Basic PSO, evidenced by larger probability of success at most dimensions and swarm sizes. For example, we can see that SPSO 2007 with swarms of size $S = 500$ has a probability of success estimate near 1 until about dimension 20, but Basic PSO’s probability of success curve for that swarm size drops from 1 around dimension 10. The size of the effects will be further quantified in later sections.

3.4.2.2 *A*-optimal Designs Generated by PSO

In Figure 3.5 we present a set of graph panels with illustrate the PSO search results for *A*-optimal designs on the $S = 50$ swarm size case. The top set of graph panels in this figure show the distributions of the PSO solutions over each run as blue points overlayed on boxplots. The vertical scale is the *A*-criterion as calculated in Borkowski (2003) [13]. Note that using the formulation in that paper, a higher *A*-score indicates a better design. Similar to the results on *D*-optimality searches, this set of graph panels generally indicates that Basic PSO and SPSO 2007 can reproduce the GA *A*-optimal design. SPSO 2011 exhibits a similar difficulty as the *D*-optimality searches in finding the *A*-optimal design for the higher dimension design scenarios. The bottom set of panels in this figure summarize the results of the PSO searches on the *A*-efficiency scale (GA optimal design taken as the comparator in this calculation). For the higher dimension scenarios Basic PSO can easily produce *A*-optimal designs with $> 80\%$ efficiency with median efficiencies near 95%. SPSO 2007 can easily produce *A*-optimal designs for the higher dimension scenarios with efficiency often $> 90\%$ and median efficiency closer to 100%. In contrast SPSO 2011 produces *A*-optimal designs with median efficiency around 85% and individuals dropping lower than 70% in some cases.

In Figure 3.6 we present a set of graph panels with illustrate the PSO search results for *A*-optimal designs on the $S = 150$ swarm size case. Generally, as compared to the $S = 50$ runs, the $S = 150$ runs exhibit higher success and efficiency in generating optimal designs. Basic PSO can be seen to easily generate *A*-optimal designs with higher than 90% efficiency for many of the higher dimensional cases and median efficiency near 95%. SPSO 2007 solutions are similarly efficient. We note that the $K = 3$ $N = 14, 15, 16$ cases appear to be seriously challenging all versions of PSO. The SPSO 2011 results improved for this larger swarm size, but the algorithm still is not

reproducing the A -optimal design.

In Figure 3.7 we present a set of graph panels which illustrate the PSO search results for A -optimal designs on the $S = 500$ swarm size case. Basic PSO seems to have benefited little from the increase in swarm size from $S = 150$ to $S = 500$ for the $K = 3$ (higher dimensional) design scenarios. This is evidenced by very similar patterns in the boxplots for these cases. In contrast, SPSO 2007 seems to achieve some benefit from the larger swarm size as it can be seen that the boxplots of the A -optimal solutions have become more dense near the GA solution for the $S = 500$ $K = 3$ cases as compared to the $S = 150$ $K = 3$ cases. SPSO 2011 does not seem to derive much benefit from the larger swarm sizes for the $K = 3$ scenarios as the distributions of the A -efficiencies for the $S = 500$ runs appear very similar to those from the $S = 150$ runs.

We present estimates of probability of success as a function of the dimension of the optimization problem (i.e., product $K * N$), PSO algorithm, and swarm size S in Figure 3.8.

This summary provides a number of clear insights, many of which are intuitive, regarding the performance of these PSO variants for generating A -optimal designs. SPSO 2007 appears to be generally more successful than Basic PSO, evidenced by larger probability of success at most dimensions and swarm sizes. The size of the effects will be further quantified in later sections. In contrast to searches for D -optimal designs, we note that Basic PSO and SPSO 2007 appeared to have some difficulty in producing the A -optimal designs for the 3 largest designs scenarios, i.e., $K = 3$, $N = 14, 15, 16$. This is evidenced by the probability of success curves being near 0 for each algorithm and each of those scenarios.

3.4.2.3 Conclusions Regarding Software Validation

The results in the previous section demonstrate that, in almost all cases, the software can produce the optimal D and A designs as generated by the genetic algorithm and published by Borkowski (2003) [13]. Further, PSO, especially Basic PSO and SPSO 2007, were demonstrated to regularly produce highly efficient D and A optimal designs for reasonable swarm sizes (e.g., $S = 100$ for up to 48 dimensional searches). Heuristic behaviors of the algorithms as described in the PSO literature and summarized in Chapter 1 were observed. For example, one would expect SPSO 2007 to achieve higher success rates because the local communication topologies were researched, developed, and demonstrated (on searches for functions with vector inputs) to achieve exactly that. Based on the information presented in the previous sections, we conclude:

1. our proposed extension of the PSO update equations to enable optimization of functions which take matrix inputs, as shown in Eq.s (3.6-3.9), is reasonable, correct, validated, and performs as expected, and
2. the data from the validation study reasonably supports the correctness of our Julia implementations of the Basic PSO, SPSO 2007 and SPSO 2011 algorithms. It is therefore reasonable to believe these functions will perform well on optimization scenarios outside the scope of this study.

In total we consider this a successful validation of the Julia PSO-OptDoE software.

In the next sections, we provide additional performance analysis, and will conclude the chapter with a discussion regarding which PSO version should be preferred for generating exact optimal designs in the hypercube. We will also comment on why SPSO 2011 appears to be behaving as it is.

3.4.3 Further Performance Analysis of the PSO Algorithms for Generating Optimal Designs

In this section we present further data and analysis from the validation study with focus on the relative performance of the three PSO algorithms. Note that as SPSO 2011 was demonstrated to not converge to the global optima in high dimensional searches, we will focus on contrasting the performance of Basic PSO with SPSO 2007 in this section. Full summary statistic tables for the study are presented in Appendix Tables 3.3 and 3.4 for the D and A optimality searches respectively.

Regarding run-time in number of iterations, we present the full distributions of number of iterations in Appendix Figures 3.17 and 3.18 for the D and A optimality searches, respectively. We extracted the median number of iterations per scenario and present the results for the D optimality searches in Figure 3.9. First, regarding Basic PSO, the graphic demonstrates that increasing the swarm size reduces the number of iterations (and appreciably for higher dimensional searches). This is likely due to Basic PSO using the global communication topology, specifically the swarm reaches consensus faster because larger swarm size requires more function evaluations at each iteration. In contrast, the graphic shows that swarm size has little effect on the median number of iterations for the SPSO 2007 algorithm. This is likely due to the SPSO 2007 using a local communication topology with expected neighborhood size of 3 particles. Thus, the swarm appears to, on average, maintain exploration for approximately the same number of iterations regardless of swarm size. This observation in conjunction with the generally higher probability of success speak favorably toward the use of SPSO 2007 for exact optimal design searches. We present a similar graph for the A optimality searches in Figure 3.13 and note a similar pattern.

We present an alternative graphic of estimates of probability of success for the D and A optimality searches in Figures 3.10 and 3.14, respectively. In these figures,

each panel represents swarm size, and the three curves in each panel correspond to probability of success estimates of the three PSO algorithms as a function of dimension. These panels make it easy to see that, for any swarm size, SPSO 2007 is the clear winner with respect to finding the global optimum. We now dive further into the analysis of probability of success between SPSO 2007 and Basic PSO.

In Figure 3.11 we present differences in probability of success (SPSO 2007 vs. Basic PSO) as a function of dimension and swarm size for the D optimality searches (95% confidence intervals on the differences are also shown where possible). For lower dimensions, say below $K * N = 20$, it appears that SPSO 2007 may have a larger probability of success for small swarm size, and this difference decreases for large swarm size indicating that SPSO 2007 and Basic PSO have similar performance for low dimension but large swarm size. This effect reverses for higher dimensions however, as the difference in probability of success increases at high dimensions (in favor of SPSO 2007) as the swarm size increases. We present a similar graphic for the A optimality searches in Figure 3.15 where these patterns are much less pronounced but the graphic is more difficult to interpret. This indicates that general statements are not possible and performance is also dependent on the objective function.

We present differences in probability of success for each algorithm as a function of swarm size and dimension for D -optimality searches in Figure 3.12. The general story told by this graphic is that Basic PSO received an appreciable increase in probability of success for lower dimensional searches when increasing the swarm size, but less so for the higher dimensional searches. In contrast, SPSO 2007 appears to benefit when increasing the swarm size by a factor of 10 for the higher dimensional searches. An analogous graphic is presented for the A optimality searches in 3.16. The general story told by this graphic is that Basic PSO seems to achieve a larger increase in success probability as the swarm size increases. For SPSO 2007 a factor of 3 increase

in swarm sizes seems to affect success probability little, but an increase is observed for a factor of 10 increase in swarm sizes.

3.5 Conclusions Regarding Feasibility of PSO to Generate Exact Optimal Designs

The results presented in this chapter are generally encouraging regarding the feasibility of using PSO to generate exact optimal experimental designs. Recall that in Chapter 2 we briefly discussed that there is no known algorithm that is guaranteed to produce the global optimal design for any scenario. Thus all algorithms, especially gradient based searches and coordinate exchange algorithms, must be run multiple times often with different starting points. Although PSO is no different in this regard, however, it is highly robust to being entrapped in local optima and was here demonstrated with high probability to produce designs with high efficiency.

Further, we learned that SPSO 2007 generally has a higher probability of success in finding the global optimum as compared to Basic PSO for the D and A optimal design searches. This is not unexpected as SPSO 2007 implements the local communication topology which was developed specifically for, and demonstrated to, increase the ability of the swarm to search the space for the global optimum (though this comes at the cost of more iterations). Last, it was interesting to observe that SPSO 2007 seemed to gain the most (with respect to probability of success) from increasing the swarm size as compared to Basic PSO.

SPSO 2011, relatively, performed rather poorly in this study. It failed to find the optimal D or A designs for the higher dimension searchers and generally did not generate designs with as high an efficiency as the Basic and SPSO 2007 algorithms. This observation is not contradictory with the performance study published by Zambrano-Bigiarini et al. (2013) [97]. In that paper it was demonstrated that SPSO 2011 as currently formulated will converge quickly to the region of the global optimum,

however, (quote) “all particles in the swarm continue to ‘fly’ over the attraction zone without converging toward the true global optimum (stagnation)”. In this paper it was demonstrated that SPSO 2011, in the sense of successfully finding the global optimum, to performed best on unimodal functions. In the set of 28 test functions to which SPSO 2011 was applied in this paper, for only about 5 test functions did the algorithm find the global optima. The authors highlighted that the introduction of the Gaussian perturbation to the velocity update equations appeared to give SPSO 2011 a sort of robustness to dimension increases without increasing the swarm size (evidenced by similar performance statistics for fixed swarm size as dimension increased). We saw the same, albeit poor, behavior.

In light of these observations and the data presented in our performance study, we recommend against the use of SPSO 2011 for optimal design searches. Ultimately, the use of SPSO 2007 for generating exact optimal designs is highly recommended.

Table 3.1: Table indicating experimental levels of the criterion, PSOVersion, and Swarm Size implemented in the validation study. The cell values indicate S , the levels of swarm size implemented in the validation study.

<u>Criterion</u>	<u>PSO Version</u>		
D	Basic	SPSO 2007	SPSO 2011
<u>Swarm Size</u> S	50	50	50
	150	150	150
	500	500	500

<u>Criterion</u>	<u>PSO Version</u>		
A	Basic	SPSO 2007	SPSO 2011
<u>Swarm Size</u> S	50	50	50
	150	150	150
	500	500	500

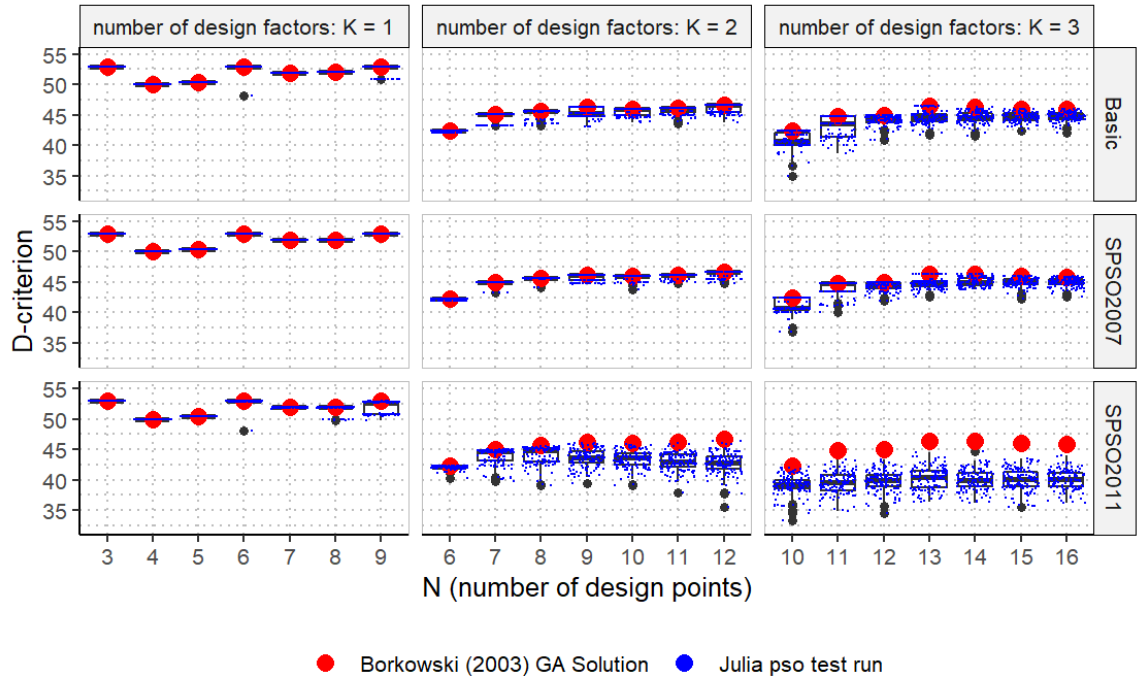
Table 3.2: Table indicating experimental levels of K (number of design factors), N (number of design points) and n_{run} (number of PSO runs implemented per design scenario).

		<u>Number of Design Factors</u>						
		$K=1$						
<u>Number of Design Points</u>		3	4	5	6	7	8	9
N								
<u>Number of PSO Runs</u>		140	140	140	140	140	140	140
n_{run}								

		$K=2$						
<u>Number of Design Points</u>		6	7	8	9	10	11	12
N								
<u>Number of PSO Runs</u>		140	140	140	140	140	140	140
n_{run}								

		$K=3$						
<u>Number of Design Points</u>		10	11	12	13	14	15	16
N								
<u>Number of PSO Runs</u>		140	140	140	140	140	140	140
n_{run}								

Distributions of D-criterion of Best Designs found by PSO
Swarm Size $S = 50$



Distributions of D-efficiency of Best Designs found by PSO
Swarm Size $S = 50$

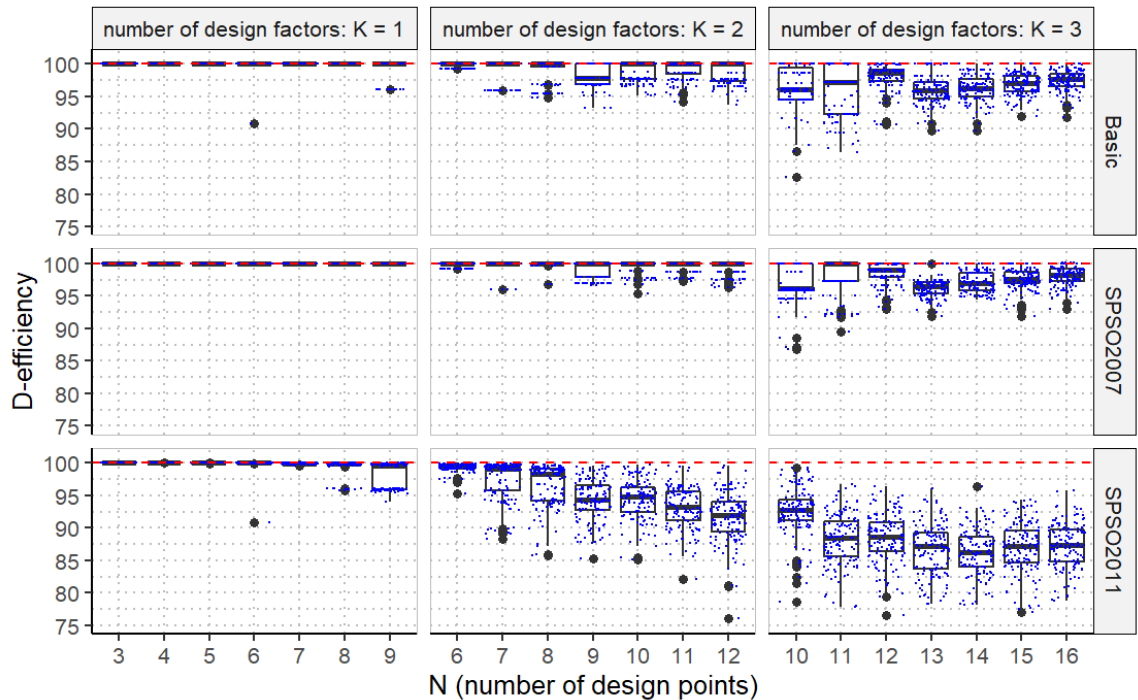
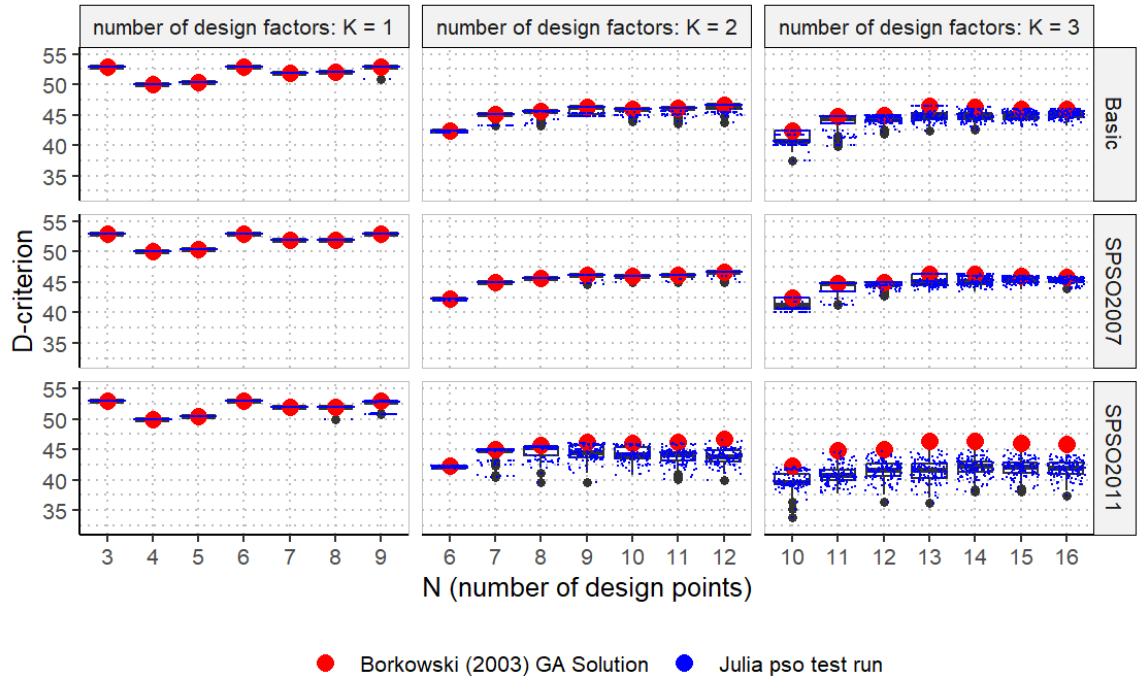


Figure 3.1: $S = 50$ runs: Boxplots of PSO D -optimal designs for each scenario (top) and corresponding D -efficiencies (bottom).

Distributions of D-criterion of Best Designs found by PSO
Swarm Size $S = 150$



Distributions of D-efficiency of Best Designs found by PSO
Swarm Size $S = 150$

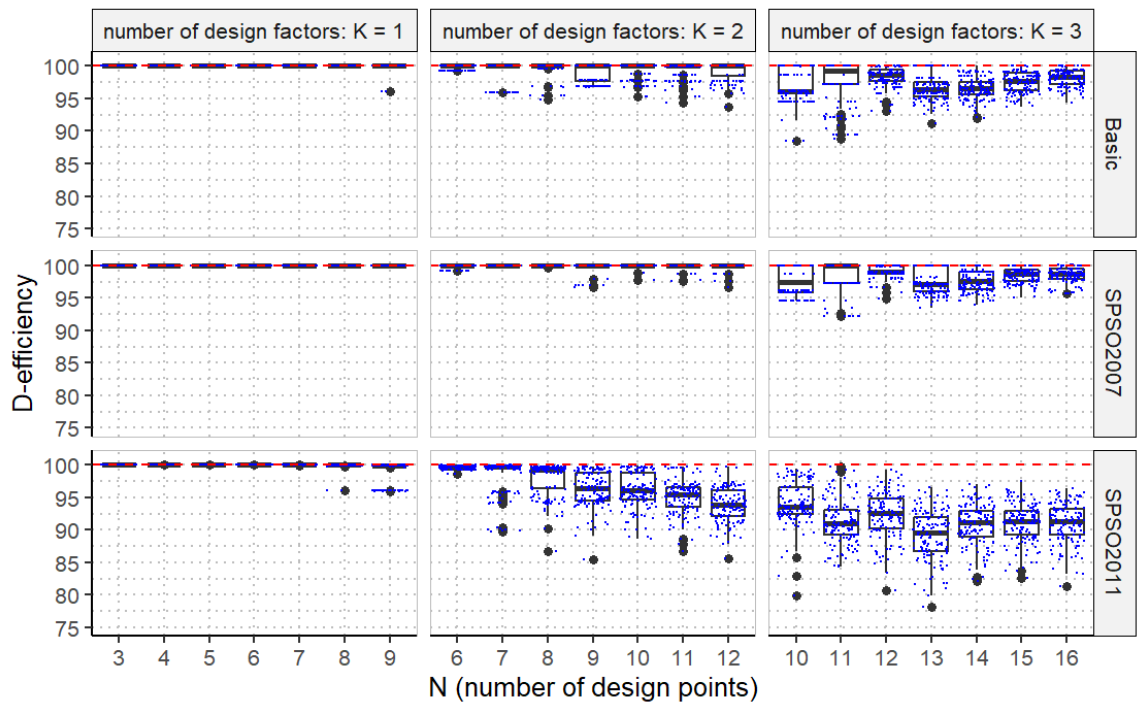


Figure 3.2: $S = 150$ runs: Boxplots of PSO D -optimal designs for each scenario (top) and corresponding D -efficiencies (bottom).

Distributions of D-criterion for Best Designs found by PSO
Swarm Size $S = 500$



Distributions of D-efficiency of Best Designs found by PSO
Swarm Size $S = 500$

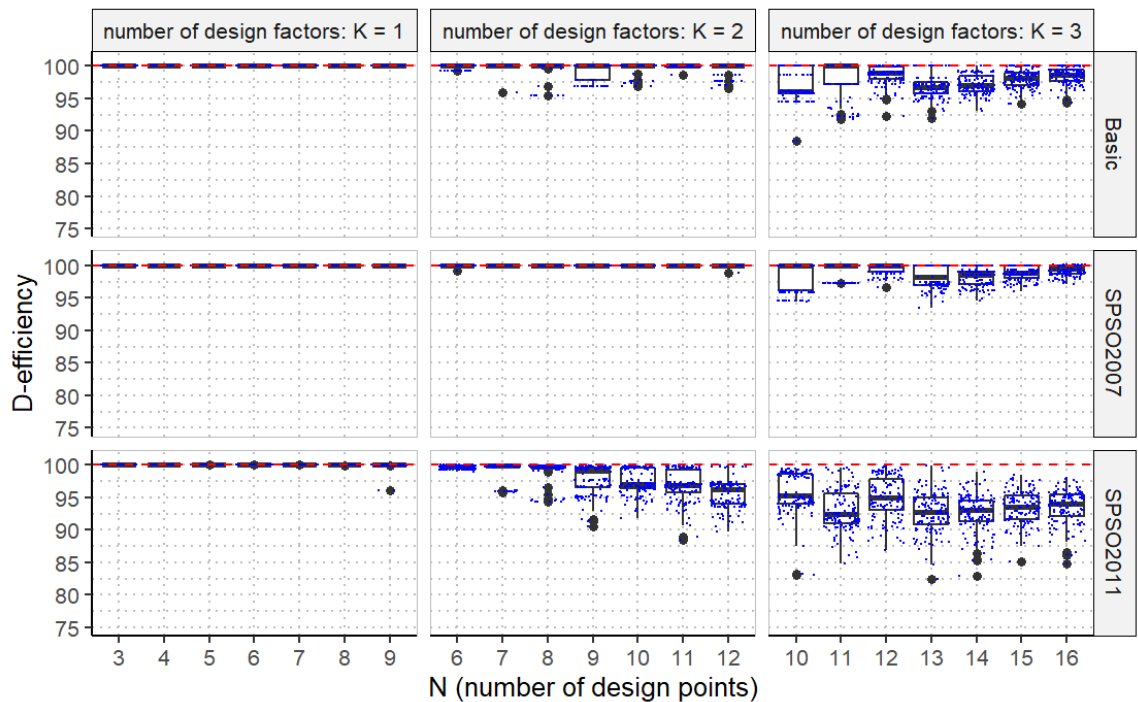


Figure 3.3: $S = 500$ runs: Boxplots of PSO D -optimal designs for each scenario (top) and corresponding D -efficiencies (bottom).

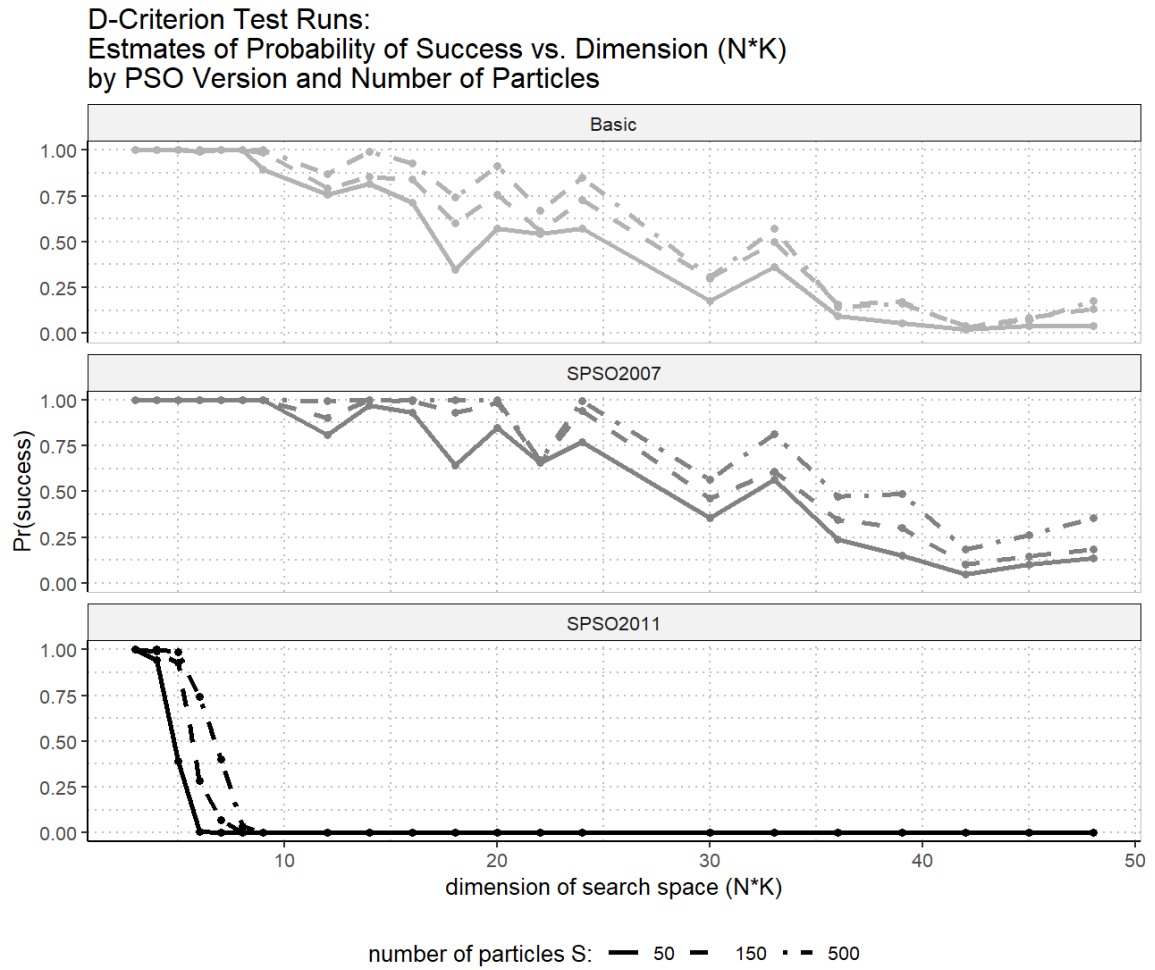


Figure 3.4: Estimates of the probability that PSO successfully finds the D -optimal design for all PSO versions, swarm sizes, and design scenarios studied.

Distributions of A-criterion of Best Designs found by PSO
Swarm Size $S = 50$



Distributions of A-efficiency of Best Designs found by PSO
Swarm Size $S = 50$

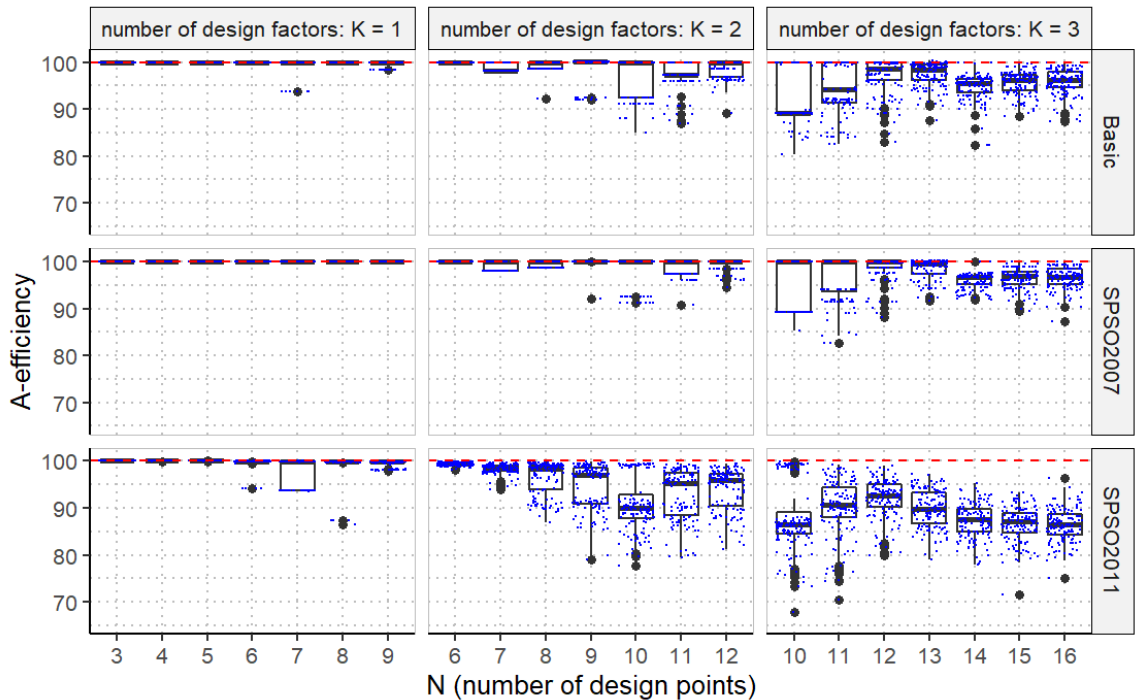


Figure 3.5: $S = 50$ runs: Boxplots of PSO A-optimal designs for each scenario (top) and corresponding A-efficiencies (bottom).

Distributions of A-criterion of Best Designs found by PSO
Swarm Size $S = 150$



Distributions of A-efficiency of Best Designs found by PSO
Swarm Size $S = 150$

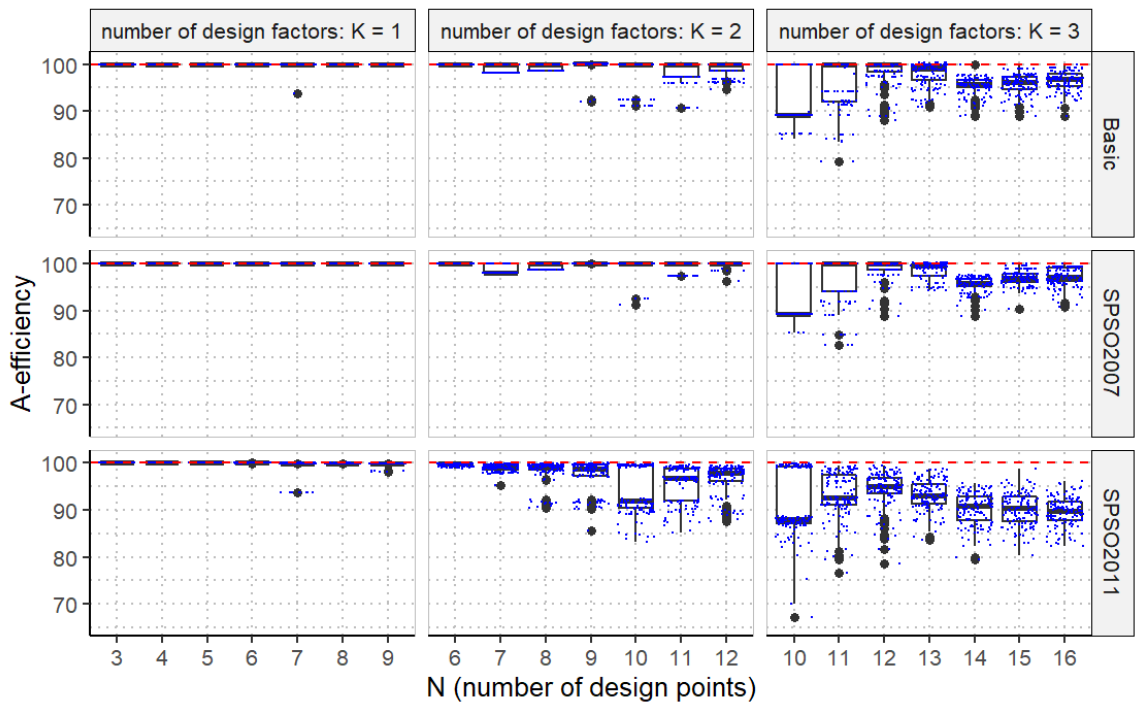
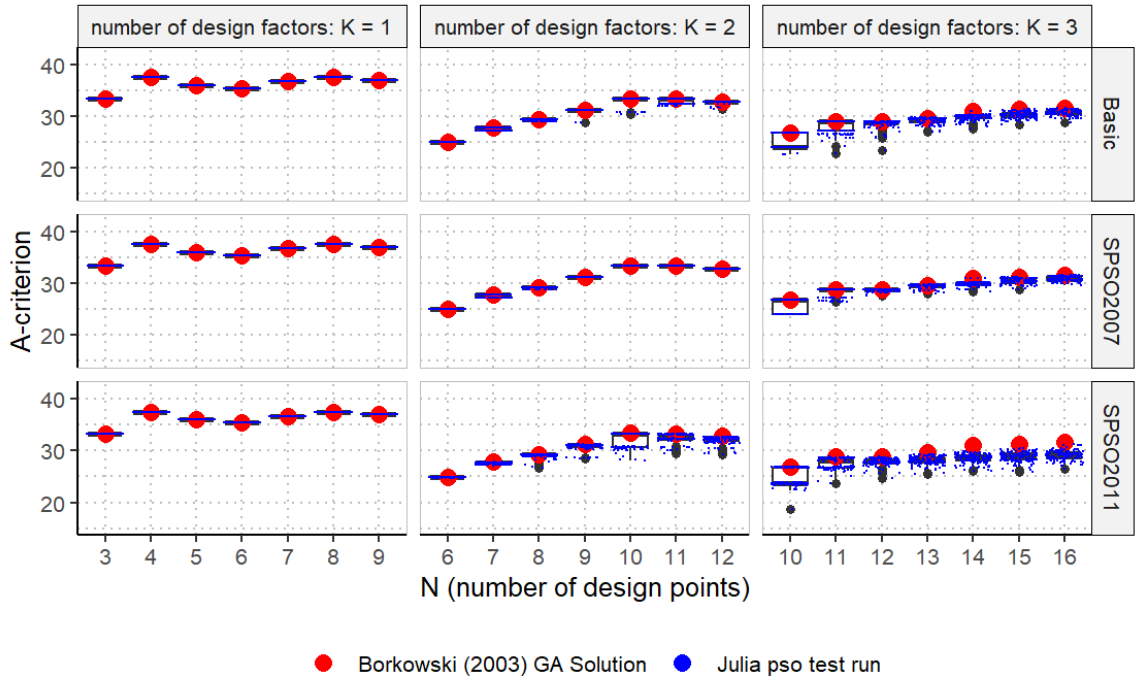


Figure 3.6: $S = 150$ runs: Boxplots of PSO A-optimal designs for each scenario (top) and corresponding A-efficiencies (bottom).

Distributions of A-criterion for Best Designs found by PSO
Swarm Size $S = 500$



Distributions of A-efficiency of Best Designs found by PSO
Swarm Size $S = 500$

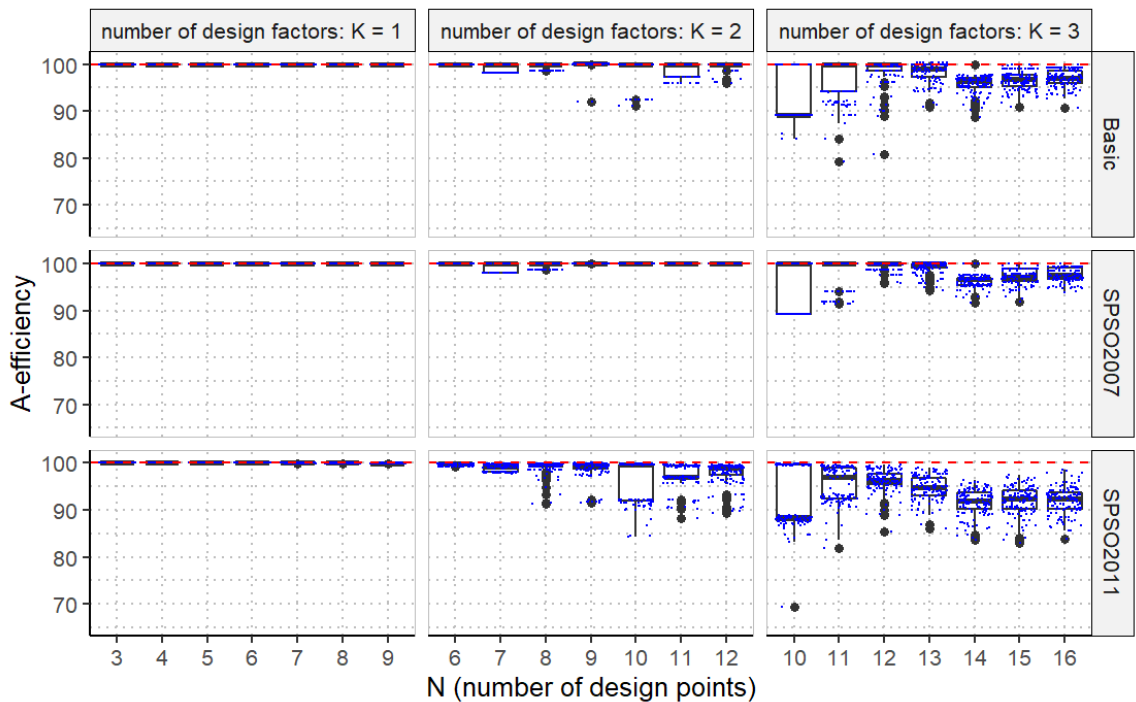


Figure 3.7: $S = 500$ runs: Boxplots of PSO A-optimal designs for each scenario (top) and corresponding A-efficiencies (bottom).

A-Criterion Test Runs:
 Estimates of Probability of Success vs. Dimension ($N \cdot K$)
 by PSO Version and Number of Particles

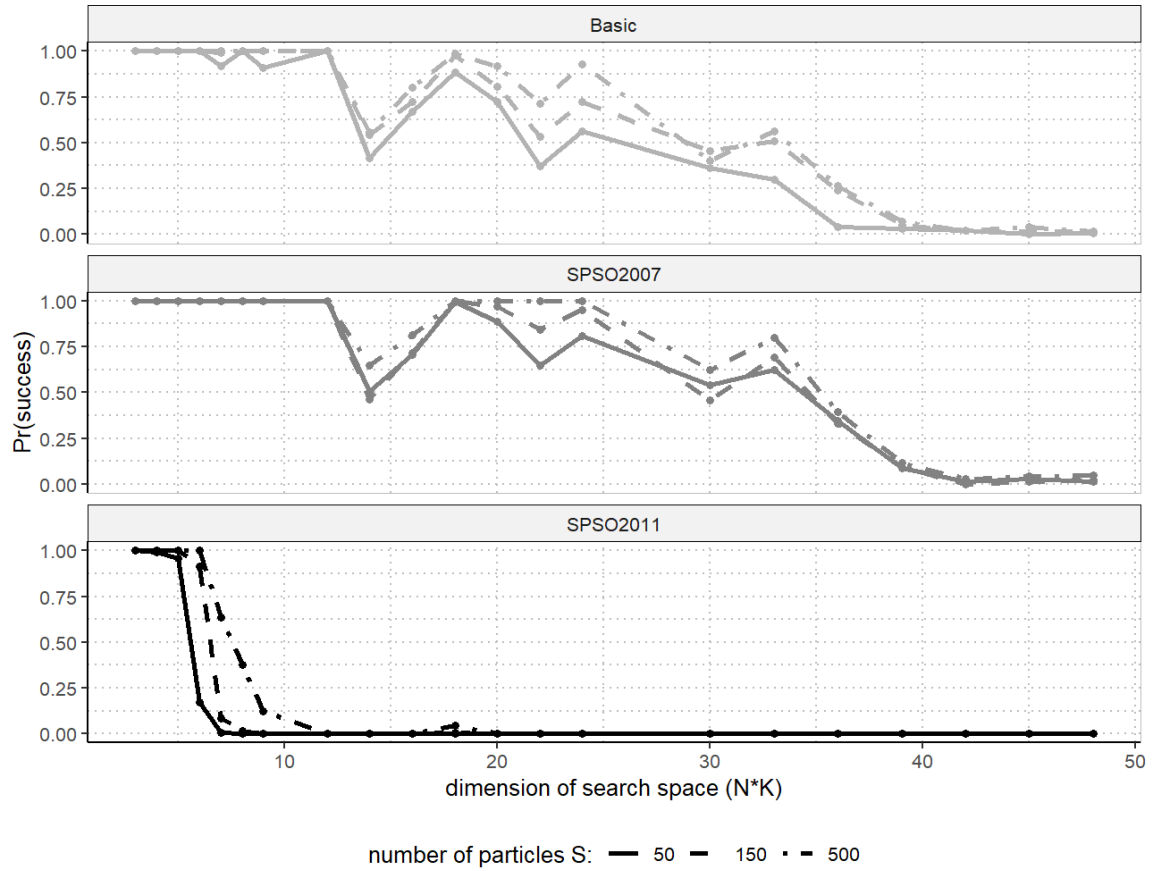


Figure 3.8: Estimates of the probability that PSO successfully finds the A -optimal design for all PSO versions, swarm sizes, and design scenarios studied.

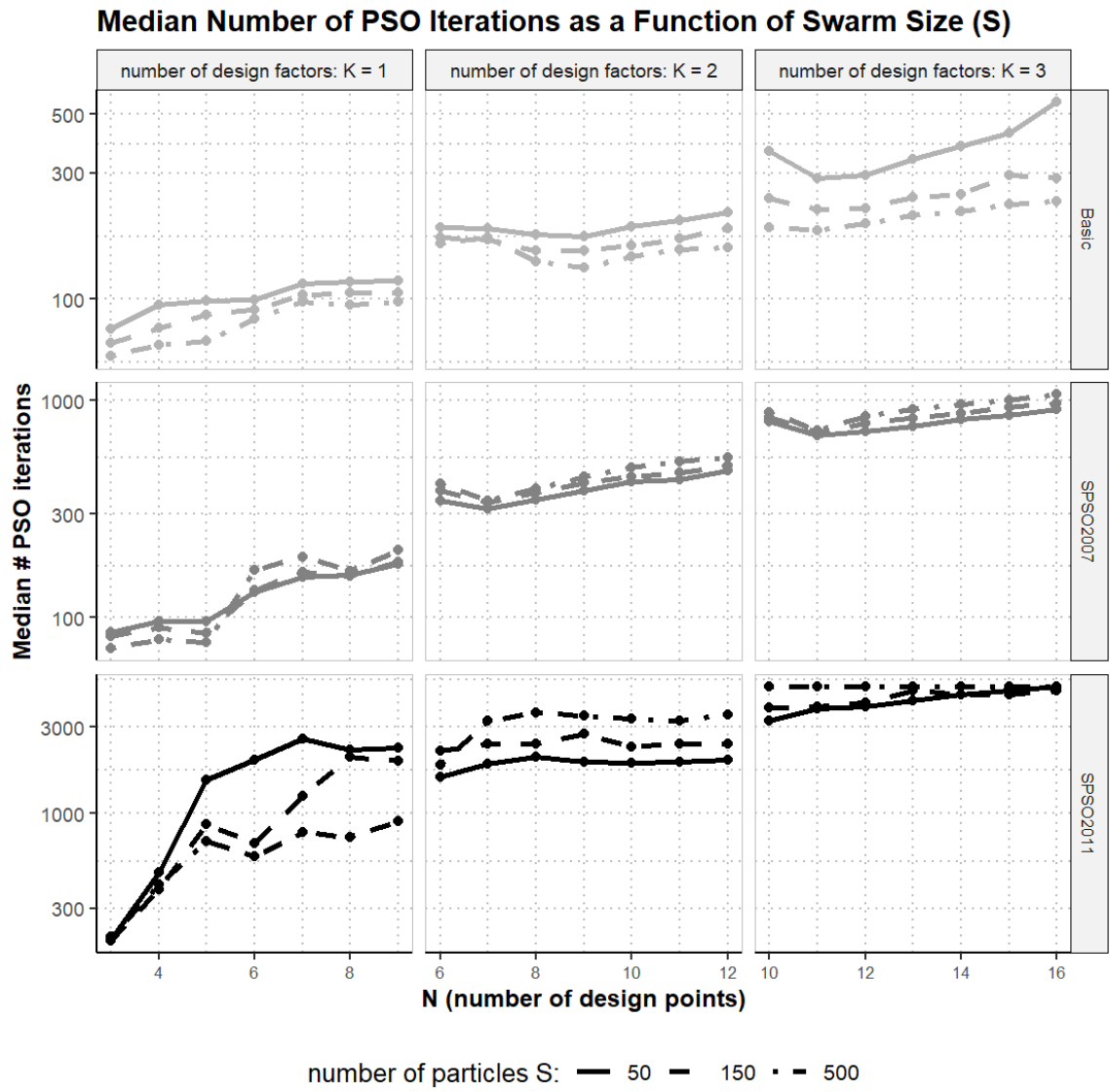


Figure 3.9: D -optimal design: Median number of iterations to algorithm terminations as a function of K , N , S , and PSO algorithm.

D-Criterion Test Runs:
Estimates of Probability of Success vs. Dimension ($N \times K$)
by Number of Particles and PSO Version

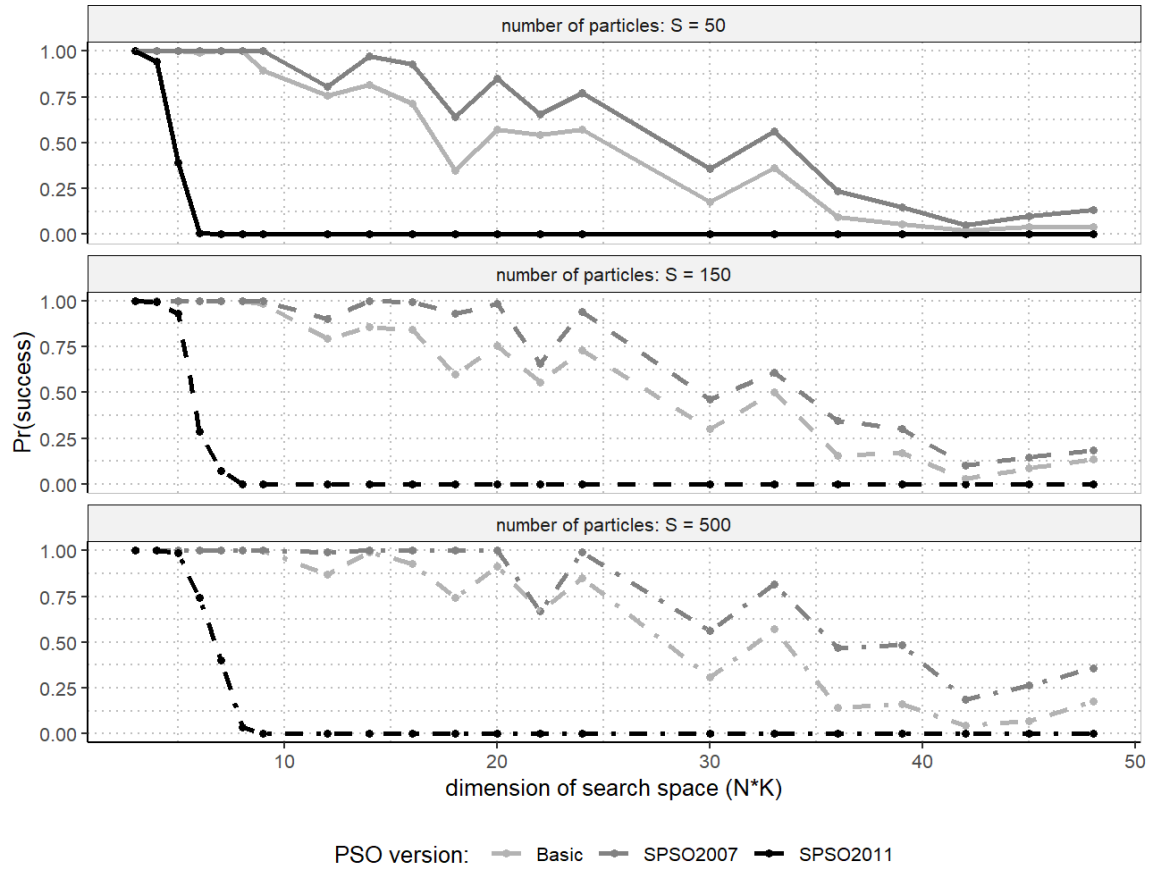


Figure 3.10: D -optimal design: probability of success estimates as a function of PSO algorithm and swarm size.

SPSO2007 - BasicPSO: Differences in Probability of Success
with 95% Confidence Intervals

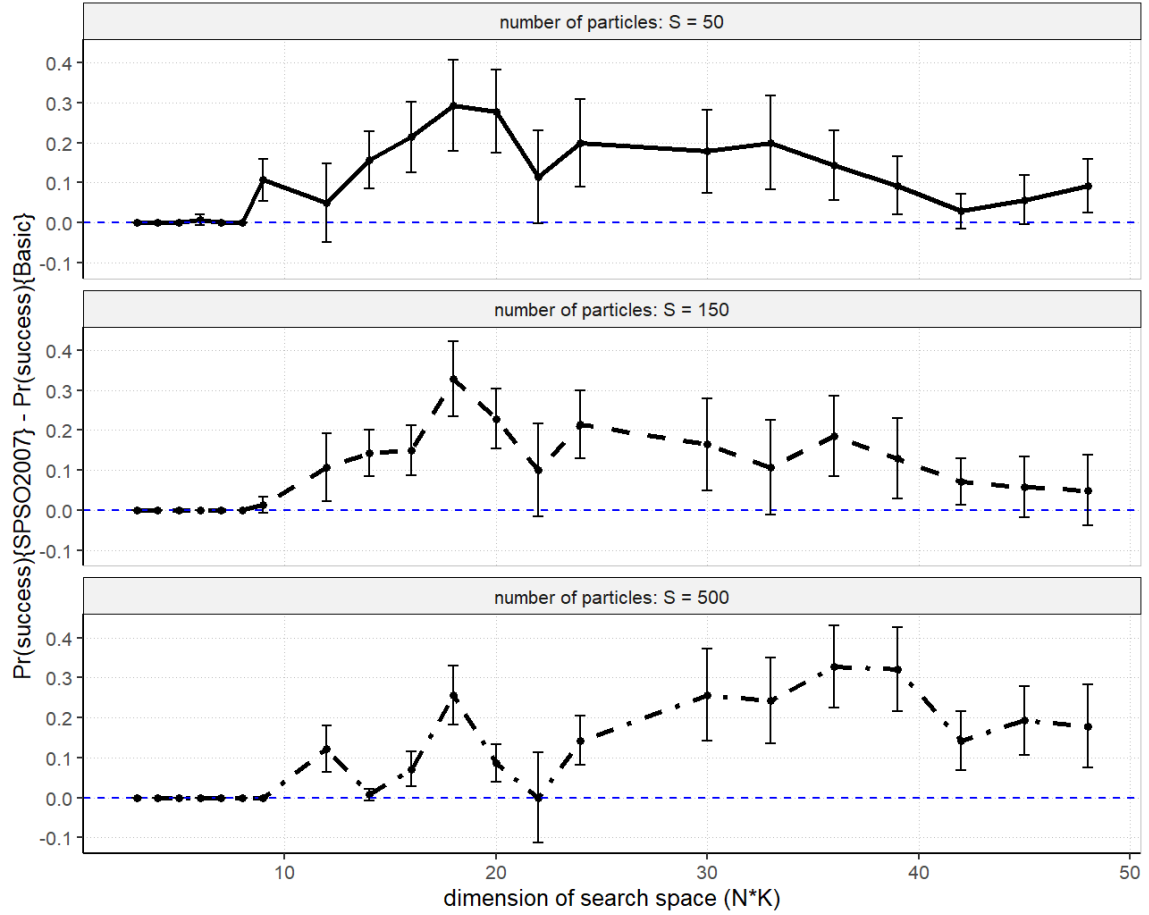


Figure 3.11: D -optimal design: comparison of increase in probability of success of SPSO 2007 compared to Basic PSO at different swarm sizes.

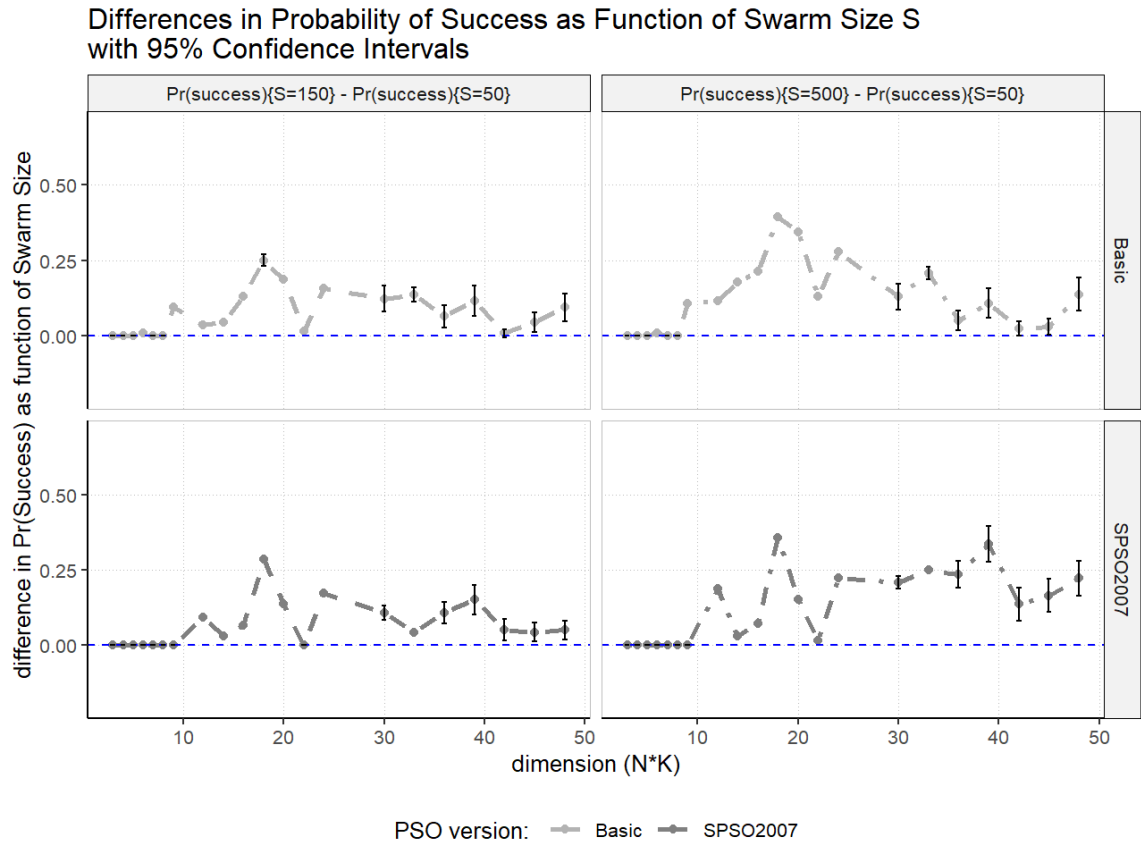


Figure 3.12: D -optimal design: comparison of increase in probability of success of each Basic PSO and SPSO 2007 attributable to increase in swarm size ($S = 50$ is basis for comparison).

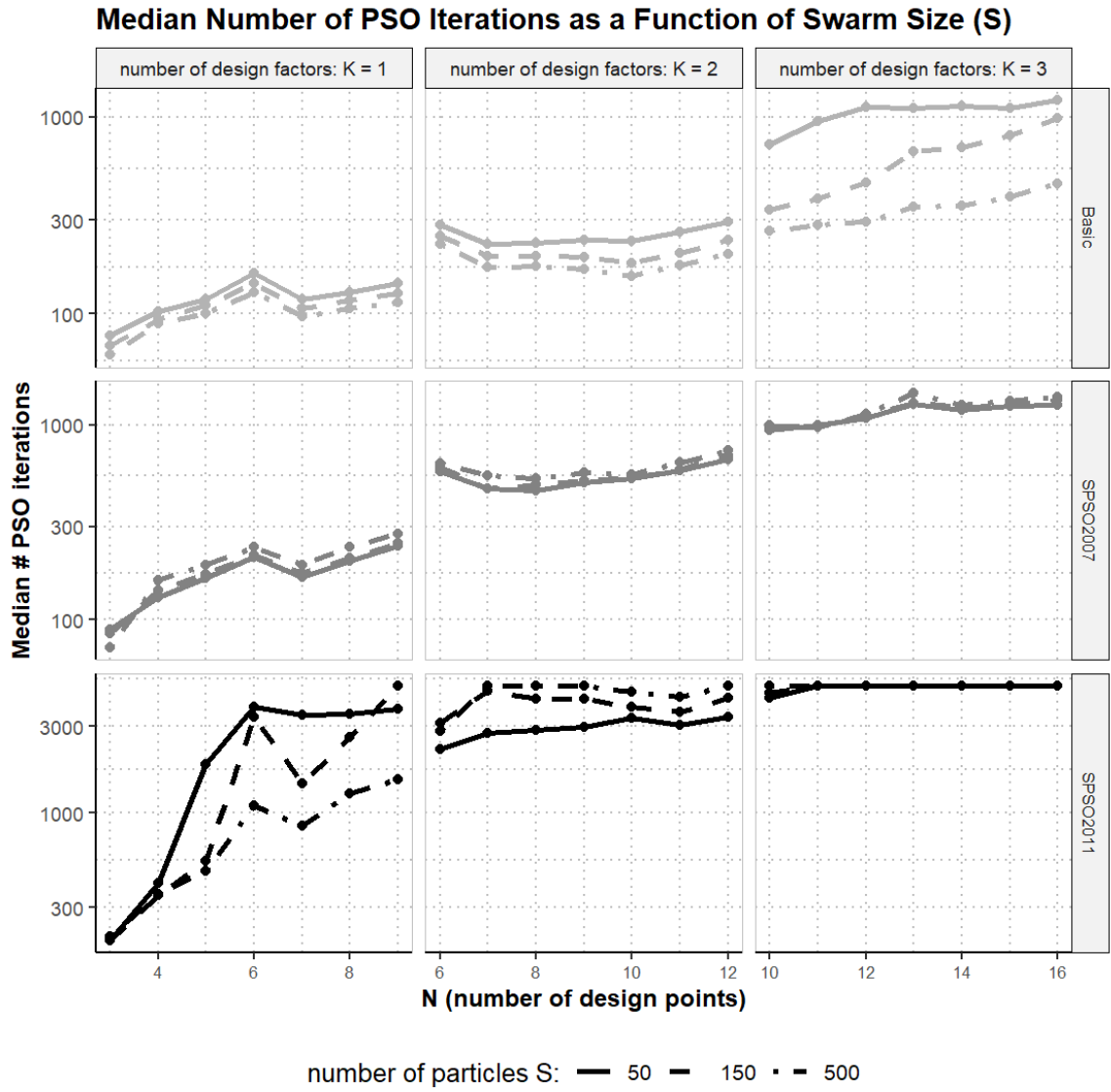


Figure 3.13: A -optimal design: Median number of iterations to algorithm terminations as a function of K , N , S , and PSO algorithm.

A-Criterion Test Runs:
 Estimates of Probability of Success vs. Dimension ($N \times K$)
 by Number of Particles and PSO Version

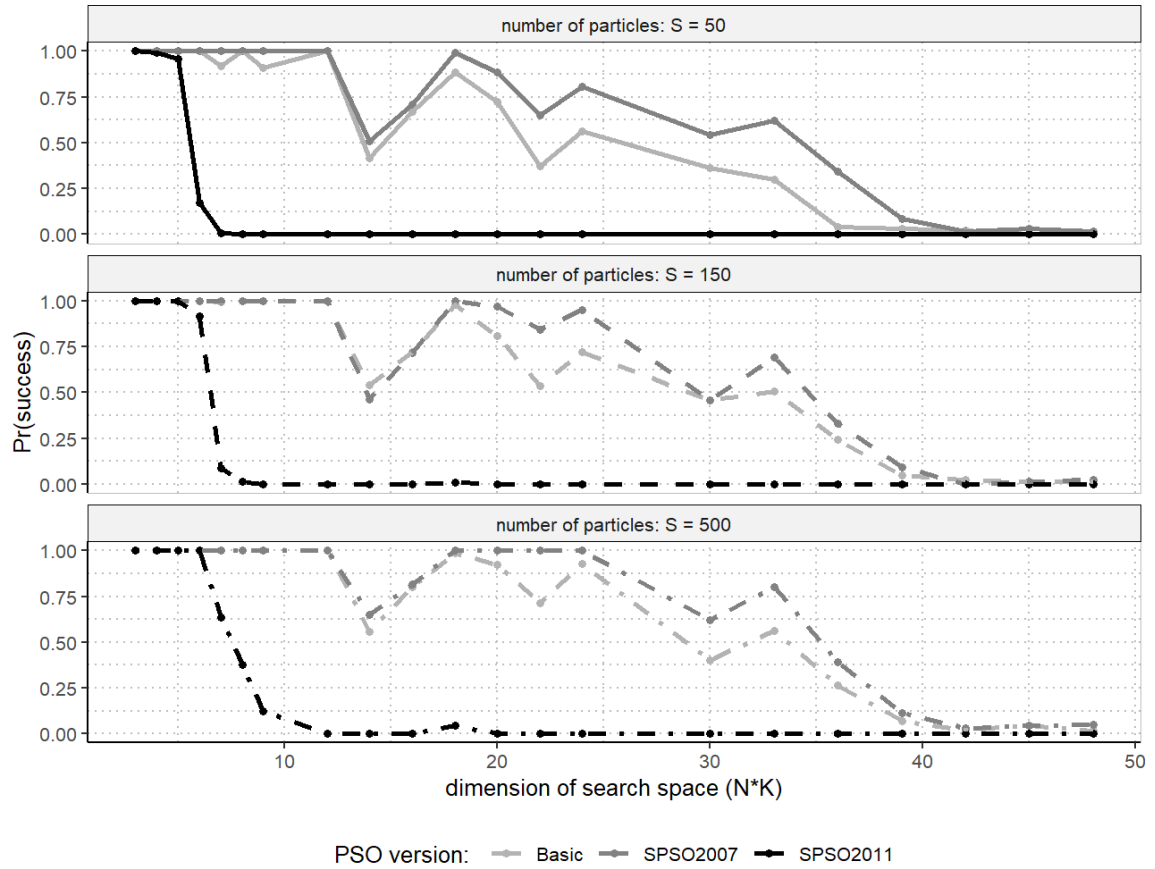


Figure 3.14: A-optimal design: probability of success estimates as a function of PSO algorithm and swarm size.

SPSO2007 - BasicPSO: Differences in Probability of Success
with 95% Confidence Intervals

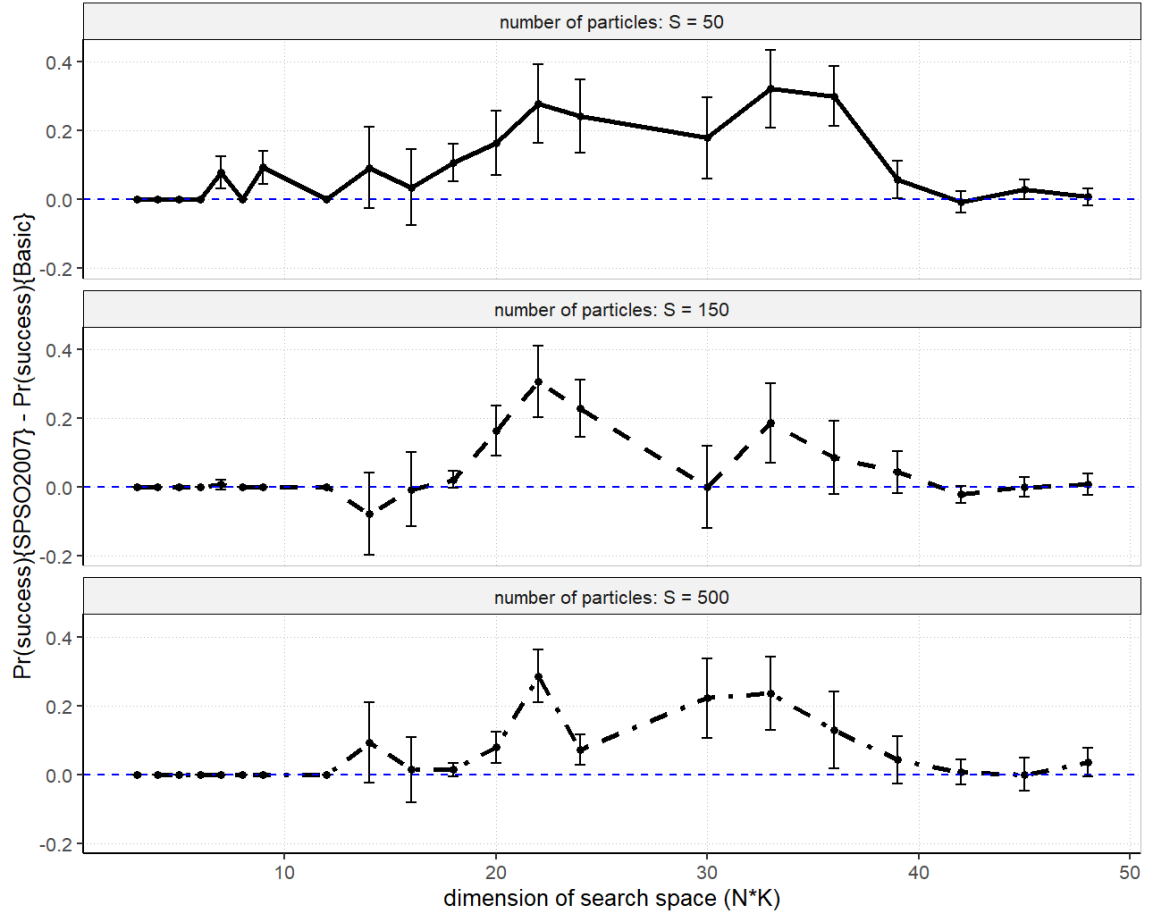


Figure 3.15: *A*-optimal design: comparison of increase in probability of success of SPSO 2007 compared to Basic PSO at different swarm sizes.

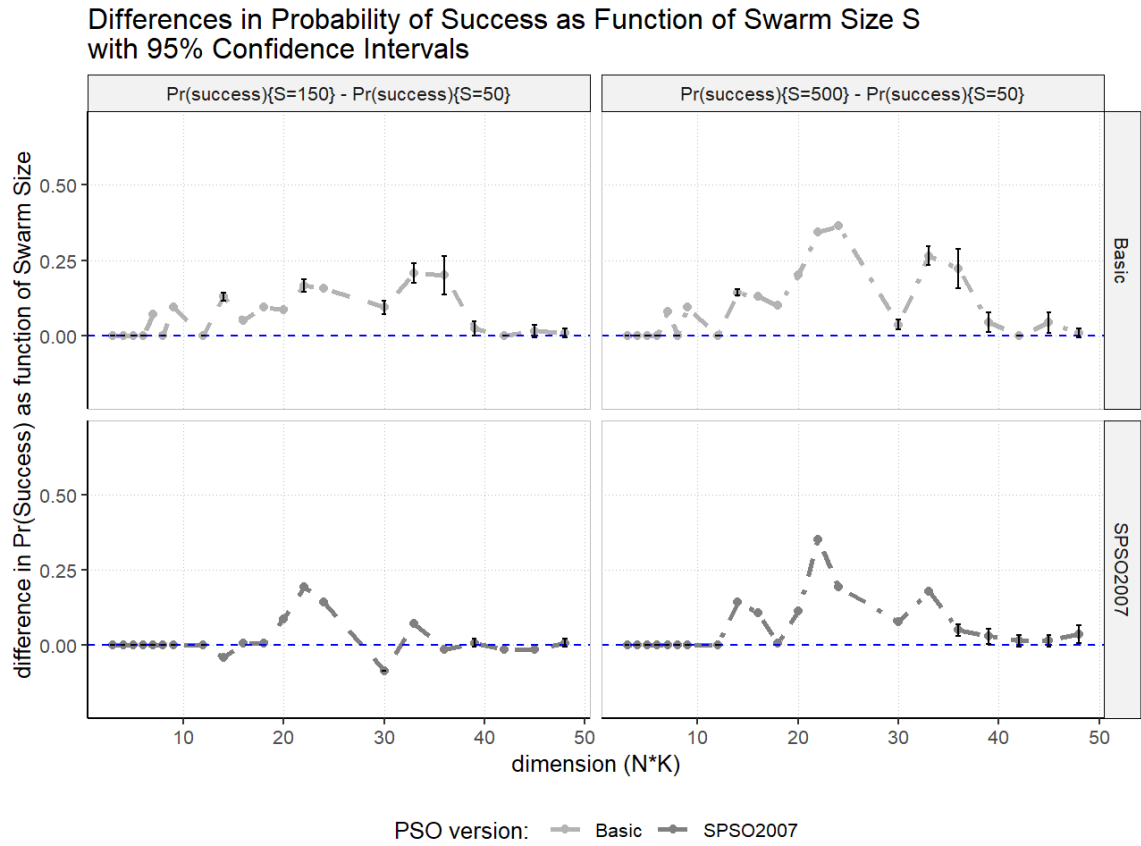


Figure 3.16: *A*-optimal design: comparison of increase in probability of success of each Basic PSO and SPSO 2007 attributable to increase in swarm size ($S = 50$ is basis for comparison).

Appendix

We provide summary statistics of the validation study in Tables 3.3 and 3.4 for the D and A optimal design searches respectively. The columns of the tables are as follows:

K: number of design factors.

N: number of design points.

S: size of swarm implemented in the search.

psover: Version of PSO algorithm being summarized in the data record.

dimension: dimension of the search $K * N$.

med.iter: the median number of iterations to algorithm termination among the 140 PSO searches.

mad.iter: the median absolute deviation of iterations to algorithm termination among the 140 PSO searches.

n.sucss: the number of the 140 PSO searches that found as good or better design as compared to the GA algorithm on the optimality criteria.

pr.sucss: estimate of the probability of success for the PSO algorithm (given swarm size and random starting points).

se.sucss: standard error of the estimate of probability of success.

lb: approximate 95% confidence lower bound on the estimate of probability of success.

ub: approximate 95% confidence upper bound on the estimate of probability of success.

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
1	3	50	Basic	3	140	77.00	8.90	140	1.00	0.00	1.00	1.00
1	3	50	SPSO 2007	3	140	85.00	13.34	140	1.00	0.00	1.00	1.00
1	3	50	SPSO 2011	3	140	202.50	37.06	140	1.00	0.00	1.00	1.00
1	3	150	Basic	3	140	68.50	8.15	140	1.00	0.00	1.00	1.00
1	3	150	SPSO 2007	3	140	82.00	14.83	140	1.00	0.00	1.00	1.00
1	3	150	SPSO 2011	3	140	199.50	32.62	140	1.00	0.00	1.00	1.00
1	3	500	Basic	3	140	61.00	7.41	140	1.00	0.00	1.00	1.00
1	3	500	SPSO 2007	3	140	72.00	13.34	140	1.00	0.00	1.00	1.00
1	3	500	SPSO 2011	3	140	208.50	37.81	140	1.00	0.00	1.00	1.00
1	4	50	Basic	4	140	95.00	14.83	140	1.00	0.00	1.00	1.00
1	4	50	SPSO 2007	4	140	96.00	11.86	140	1.00	0.00	1.00	1.00
1	4	50	SPSO 2011	4	140	474.50	181.62	132	0.94	0.02	0.90	0.98
1	4	150	Basic	4	140	78.00	11.86	140	1.00	0.00	1.00	1.00
1	4	150	SPSO 2007	4	140	89.50	14.08	140	1.00	0.00	1.00	1.00
1	4	150	SPSO 2011	4	140	382.50	57.08	139	0.99	0.01	0.98	1.00
1	4	500	Basic	4	140	67.00	11.86	140	1.00	0.00	1.00	1.00
1	4	500	SPSO 2007	4	140	79.50	11.12	140	1.00	0.00	1.00	1.00
1	4	500	SPSO 2011	4	140	405.00	60.05	140	1.00	0.00	1.00	1.00
1	5	50	Basic	5	140	98.50	11.12	140	1.00	0.00	1.00	1.00
1	5	50	SPSO 2007	5	140	95.50	20.76	140	1.00	0.00	1.00	1.00
1	5	50	SPSO 2011	5	140	1522.00	1606.40	55	0.39	0.04	0.31	0.48
1	5	150	Basic	5	140	87.00	13.34	140	1.00	0.00	1.00	1.00
1	5	150	SPSO 2007	5	140	84.00	14.83	140	1.00	0.00	1.00	1.00
1	5	150	SPSO 2011	5	140	872.50	353.60	130	0.93	0.02	0.89	0.97

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
1	5	500	Basic	5	140	69.50	20.76	140	1.00	0.00	1.00	1.00
1	5	500	SPSO 2007	5	140	77.00	11.86	140	1.00	0.00	1.00	1.00
1	5	500	SPSO 2011	5	140	702.00	124.54	138	0.99	0.01	0.97	1.00
1	6	50	Basic	6	140	99.00	6.67	139	0.99	0.01	0.98	1.00
1	6	50	SPSO 2007	6	140	131.00	14.83	140	1.00	0.00	1.00	1.00
1	6	50	SPSO 2011	6	140	1976.50	2013.37	1	0.01	0.01	0.00	0.02
1	6	150	Basic	6	140	91.00	4.45	140	1.00	0.00	1.00	1.00
1	6	150	SPSO 2007	6	140	133.00	14.83	140	1.00	0.00	1.00	1.00
1	6	150	SPSO 2011	6	140	686.00	467.02	40	0.29	0.04	0.21	0.36
1	6	500	Basic	6	140	84.00	4.45	140	1.00	0.00	1.00	1.00
1	6	500	SPSO 2007	6	140	166.00	22.24	140	1.00	0.00	1.00	1.00
1	6	500	SPSO 2011	6	140	579.00	304.67	104	0.74	0.04	0.67	0.82
1	7	50	Basic	7	140	114.00	7.41	140	1.00	0.00	1.00	1.00
1	7	50	SPSO 2007	7	140	154.00	17.79	140	1.00	0.00	1.00	1.00
1	7	50	SPSO 2011	7	140	2577.00	1866.59	0	0.00	0.00	0.00	0.00
1	7	150	Basic	7	140	104.00	5.93	140	1.00	0.00	1.00	1.00
1	7	150	SPSO 2007	7	140	162.00	22.98	140	1.00	0.00	1.00	1.00
1	7	150	SPSO 2011	7	140	1241.00	1130.48	10	0.07	0.02	0.03	0.11
1	7	500	Basic	7	140	97.50	6.67	140	1.00	0.00	1.00	1.00
1	7	500	SPSO 2007	7	140	189.00	26.69	140	1.00	0.00	1.00	1.00
1	7	500	SPSO 2011	7	140	791.50	538.93	56	0.40	0.04	0.32	0.48
1	8	50	Basic	8	140	116.00	7.41	140	1.00	0.00	1.00	1.00
1	8	50	SPSO 2007	8	140	156.00	27.43	140	1.00	0.00	1.00	1.00
1	8	50	SPSO 2011	8	140	2230.50	1122.33	0	0.00	0.00	0.00	0.00
1	8	150	Basic	8	140	106.00	7.41	140	1.00	0.00	1.00	1.00
1	8	150	SPSO 2007	8	140	155.00	28.17	140	1.00	0.00	1.00	1.00
1	8	150	SPSO 2011	8	140	2043.50	2372.90	0	0.00	0.00	0.00	0.00

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
1	8	500	Basic	8	140	95.50	8.15	140	1.00	0.00	1.00	1.00
1	8	500	SPSO 2007	8	140	163.50	25.95	140	1.00	0.00	1.00	1.00
1	8	500	SPSO 2011	8	140	741.00	455.90	5	0.04	0.02	0.00	0.07
1	9	50	Basic	9	140	117.00	7.41	125	0.89	0.03	0.84	0.95
1	9	50	SPSO 2007	9	140	177.00	20.76	140	1.00	0.00	1.00	1.00
1	9	50	SPSO 2011	9	140	2294.50	1123.81	0	0.00	0.00	0.00	0.00
1	9	150	Basic	9	140	106.00	4.45	138	0.99	0.01	0.97	1.00
1	9	150	SPSO 2007	9	140	180.00	20.76	140	1.00	0.00	1.00	1.00
1	9	150	SPSO 2011	9	140	1959.00	2048.21	0	0.00	0.00	0.00	0.00
1	9	500	Basic	9	140	98.00	4.45	140	1.00	0.00	1.00	1.00
1	9	500	SPSO 2007	9	140	204.50	30.39	140	1.00	0.00	1.00	1.00
1	9	500	SPSO 2011	9	140	902.00	663.46	0	0.00	0.00	0.00	0.00
2	6	50	Basic	12	140	186.00	35.58	106	0.76	0.04	0.68	0.83
2	6	50	SPSO 2007	12	140	344.00	72.65	113	0.81	0.03	0.74	0.87
2	6	50	SPSO 2011	12	140	1599.00	661.98	0	0.00	0.00	0.00	0.00
2	6	150	Basic	12	140	171.00	36.32	111	0.79	0.03	0.72	0.86
2	6	150	SPSO 2007	12	140	382.50	63.75	126	0.90	0.03	0.85	0.95
2	6	150	SPSO 2011	12	140	2220.50	1160.88	0	0.00	0.00	0.00	0.00
2	6	500	Basic	12	140	162.50	25.95	122	0.87	0.03	0.81	0.93
2	6	500	SPSO 2007	12	140	411.50	65.98	139	0.99	0.01	0.98	1.00
2	6	500	SPSO 2011	12	140	1856.00	1770.97	0	0.00	0.00	0.00	0.00
2	7	50	Basic	14	140	184.50	37.81	114	0.81	0.03	0.75	0.88
2	7	50	SPSO 2007	14	140	316.00	59.30	136	0.97	0.01	0.94	1.00
2	7	50	SPSO 2011	14	140	1877.50	772.43	0	0.00	0.00	0.00	0.00
2	7	150	Basic	14	140	168.00	38.55	120	0.86	0.03	0.80	0.92
2	7	150	SPSO 2007	14	140	336.00	57.82	140	1.00	0.00	1.00	1.00
2	7	150	SPSO 2011	14	140	2430.50	1055.61	0	0.00	0.00	0.00	0.00

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
2	7	500	Basic	14	140	170.50	37.81	139	0.99	0.01	0.98	1.00
2	7	500	SPSO 2007	14	140	345.00	49.67	140	1.00	0.00	1.00	1.00
2	7	500	SPSO 2011	14	140	3252.50	1476.67	0	0.00	0.00	0.00	0.00
2	8	50	Basic	16	140	175.00	32.62	100	0.71	0.04	0.64	0.79
2	8	50	SPSO 2007	16	140	350.00	56.34	130	0.93	0.02	0.89	0.97
2	8	50	SPSO 2011	16	140	2040.50	1063.02	0	0.00	0.00	0.00	0.00
2	8	150	Basic	16	140	153.00	20.76	118	0.84	0.03	0.78	0.90
2	8	150	SPSO 2007	16	140	376.50	54.11	139	0.99	0.01	0.98	1.00
2	8	150	SPSO 2011	16	140	2431.50	985.93	0	0.00	0.00	0.00	0.00
2	8	500	Basic	16	140	139.00	16.31	130	0.93	0.02	0.89	0.97
2	8	500	SPSO 2007	16	140	393.00	47.44	140	1.00	0.00	1.00	1.00
2	8	500	SPSO 2011	16	140	3578.50	1490.75	0	0.00	0.00	0.00	0.00
2	9	50	Basic	18	140	173.00	20.02	49	0.35	0.04	0.27	0.43
2	9	50	SPSO 2007	18	140	385.50	61.53	90	0.64	0.04	0.56	0.72
2	9	50	SPSO 2011	18	140	1938.00	779.85	0	0.00	0.00	0.00	0.00
2	9	150	Basic	18	140	152.00	19.27	84	0.60	0.04	0.52	0.68
2	9	150	SPSO 2007	18	140	417.00	60.05	130	0.93	0.02	0.89	0.97
2	9	150	SPSO 2011	18	140	2748.00	1387.71	0	0.00	0.00	0.00	0.00
2	9	500	Basic	18	140	131.50	11.12	104	0.74	0.04	0.67	0.82
2	9	500	SPSO 2007	18	140	445.00	44.48	140	1.00	0.00	1.00	1.00
2	9	500	SPSO 2011	18	140	3481.50	1594.54	0	0.00	0.00	0.00	0.00
2	10	50	Basic	20	140	188.00	28.17	80	0.57	0.04	0.49	0.66
2	10	50	SPSO 2007	20	140	423.00	71.91	119	0.85	0.03	0.79	0.91
2	10	50	SPSO 2011	20	140	1895.50	524.10	0	0.00	0.00	0.00	0.00
2	10	150	Basic	20	140	159.00	16.31	106	0.76	0.04	0.68	0.83
2	10	150	SPSO 2007	20	140	447.00	63.75	138	0.99	0.01	0.97	1.00
2	10	150	SPSO 2011	20	140	2342.50	744.27	0	0.00	0.00	0.00	0.00

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
2	10	500	Basic	20	140	145.00	17.79	128	0.91	0.02	0.87	0.96
2	10	500	SPSO 2007	20	140	489.00	63.01	140	1.00	0.00	1.00	1.00
2	10	500	SPSO 2011	20	140	3352.00	1208.32	0	0.00	0.00	0.00	0.00
2	11	50	Basic	22	140	198.50	33.36	76	0.54	0.04	0.46	0.63
2	11	50	SPSO 2007	22	140	433.00	63.01	92	0.66	0.04	0.58	0.74
2	11	50	SPSO 2011	22	140	1936.00	584.89	0	0.00	0.00	0.00	0.00
2	11	150	Basic	22	140	170.00	27.43	78	0.56	0.04	0.47	0.64
2	11	150	SPSO 2007	22	140	466.50	77.84	92	0.66	0.04	0.58	0.74
2	11	150	SPSO 2011	22	140	2435.50	701.27	0	0.00	0.00	0.00	0.00
2	11	500	Basic	22	140	154.50	25.20	94	0.67	0.04	0.59	0.75
2	11	500	SPSO 2007	22	140	520.50	68.94	94	0.67	0.04	0.59	0.75
2	11	500	SPSO 2011	22	140	3251.00	978.52	0	0.00	0.00	0.00	0.00
2	12	50	Basic	24	140	213.00	36.32	80	0.57	0.04	0.49	0.66
2	12	50	SPSO 2007	24	140	476.50	78.58	108	0.77	0.04	0.70	0.84
2	12	50	SPSO 2011	24	140	1970.50	583.40	0	0.00	0.00	0.00	0.00
2	12	150	Basic	24	140	185.50	29.65	102	0.73	0.04	0.65	0.80
2	12	150	SPSO 2007	24	140	500.50	66.72	132	0.94	0.02	0.90	0.98
2	12	150	SPSO 2011	24	140	2429.50	657.53	0	0.00	0.00	0.00	0.00
2	12	500	Basic	24	140	157.00	20.76	119	0.85	0.03	0.79	0.91
2	12	500	SPSO 2007	24	140	548.00	59.30	139	0.99	0.01	0.98	1.00
2	12	500	SPSO 2011	24	140	3499.00	1177.93	0	0.00	0.00	0.00	0.00
3	10	50	Basic	30	140	361.50	190.51	25	0.18	0.03	0.11	0.24
3	10	50	SPSO 2007	30	140	797.00	128.24	50	0.36	0.04	0.28	0.44
3	10	50	SPSO 2011	30	140	3229.00	1203.13	0	0.00	0.00	0.00	0.00
3	10	150	Basic	30	140	241.50	57.08	42	0.30	0.04	0.22	0.38
3	10	150	SPSO 2007	30	140	839.50	123.06	65	0.46	0.04	0.38	0.55
3	10	150	SPSO 2011	30	140	3850.50	1704.25	0	0.00	0.00	0.00	0.00

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
3	10	500	Basic	30	140	187.00	32.62	43	0.31	0.04	0.23	0.39
3	10	500	SPSO 2007	30	140	883.50	114.90	79	0.56	0.04	0.48	0.65
3	10	500	SPSO 2011	30	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	11	50	Basic	33	140	285.50	83.77	51	0.36	0.04	0.28	0.45
3	11	50	SPSO 2007	33	140	691.50	96.37	79	0.56	0.04	0.48	0.65
3	11	50	SPSO 2011	33	140	3799.00	1753.17	0	0.00	0.00	0.00	0.00
3	11	150	Basic	33	140	217.50	35.58	70	0.50	0.04	0.42	0.58
3	11	150	SPSO 2007	33	140	710.50	76.35	85	0.61	0.04	0.52	0.69
3	11	150	SPSO 2011	33	140	3884.00	1654.58	0	0.00	0.00	0.00	0.00
3	11	500	Basic	33	140	182.00	31.13	80	0.57	0.04	0.49	0.66
3	11	500	SPSO 2007	33	140	729.00	85.99	114	0.81	0.03	0.75	0.88
3	11	500	SPSO 2011	33	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	12	50	Basic	36	140	293.50	91.92	13	0.09	0.02	0.04	0.14
3	12	50	SPSO 2007	36	140	723.50	99.33	33	0.24	0.04	0.16	0.31
3	12	50	SPSO 2011	36	140	3900.00	1507.80	0	0.00	0.00	0.00	0.00
3	12	150	Basic	36	140	220.50	43.00	22	0.16	0.03	0.10	0.22
3	12	150	SPSO 2007	36	140	784.50	84.51	48	0.34	0.04	0.26	0.42
3	12	150	SPSO 2011	36	140	4083.50	1358.80	0	0.00	0.00	0.00	0.00
3	12	500	Basic	36	140	193.00	32.62	20	0.14	0.03	0.08	0.20
3	12	500	SPSO 2007	36	140	846.50	93.40	66	0.47	0.04	0.39	0.56
3	12	500	SPSO 2011	36	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	13	50	Basic	39	140	339.50	134.92	8	0.06	0.02	0.02	0.10
3	13	50	SPSO 2007	39	140	758.50	97.85	21	0.15	0.03	0.09	0.21
3	13	50	SPSO 2011	39	140	4206.00	1177.18	0	0.00	0.00	0.00	0.00
3	13	150	Basic	39	140	242.00	48.18	24	0.17	0.03	0.11	0.24
3	13	150	SPSO 2007	39	140	826.00	85.99	42	0.30	0.04	0.22	0.38
3	13	150	SPSO 2011	39	140	4761.00	354.34	0	0.00	0.00	0.00	0.00

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
3	13	500	Basic	39	140	206.50	35.58	23	0.16	0.03	0.10	0.23
3	13	500	SPSO 2007	39	140	910.50	95.63	68	0.49	0.04	0.40	0.57
3	13	500	SPSO 2011	39	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	14	50	Basic	42	140	379.50	154.19	3	0.02	0.01	0.00	0.05
3	14	50	SPSO 2007	42	140	820.00	75.61	7	0.05	0.02	0.01	0.09
3	14	50	SPSO 2011	42	140	4523.00	707.20	0	0.00	0.00	0.00	0.00
3	14	150	Basic	42	140	250.00	48.18	4	0.03	0.01	0.00	0.06
3	14	150	SPSO 2007	42	140	875.00	89.70	14	0.10	0.03	0.05	0.15
3	14	150	SPSO 2011	42	140	4546.00	673.10	0	0.00	0.00	0.00	0.00
3	14	500	Basic	42	140	214.00	43.00	6	0.04	0.02	0.01	0.08
3	14	500	SPSO 2007	42	140	962.00	105.26	26	0.19	0.03	0.12	0.25
3	14	500	SPSO 2011	42	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	15	50	Basic	45	140	424.00	204.60	6	0.04	0.02	0.01	0.08
3	15	50	SPSO 2007	45	140	851.50	100.08	14	0.10	0.03	0.05	0.15
3	15	50	SPSO 2011	45	140	4754.50	363.98	0	0.00	0.00	0.00	0.00
3	15	150	Basic	45	140	293.00	65.23	12	0.09	0.02	0.04	0.13
3	15	150	SPSO 2007	45	140	934.50	98.59	20	0.14	0.03	0.08	0.20
3	15	150	SPSO 2011	45	140	4514.50	719.80	0	0.00	0.00	0.00	0.00
3	15	500	Basic	45	140	229.00	40.03	10	0.07	0.02	0.03	0.11
3	15	500	SPSO 2007	45	140	1004.50	94.89	37	0.26	0.04	0.19	0.34
3	15	500	SPSO 2011	45	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	16	50	Basic	48	140	555.50	290.59	6	0.04	0.02	0.01	0.08
3	16	50	SPSO 2007	48	140	907.50	128.99	19	0.14	0.03	0.08	0.19
3	16	50	SPSO 2011	48	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	16	150	Basic	48	140	286.00	69.68	19	0.14	0.03	0.08	0.19
3	16	150	SPSO 2007	48	140	969.50	109.71	26	0.19	0.03	0.12	0.25
3	16	150	SPSO 2011	48	140	4765.00	348.41	0	0.00	0.00	0.00	0.00

Table 3.3: Summary statistics from the validation study for PSO runs in generating D -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
3	16	500	Basic	48	140	234.50	47.44	25	0.18	0.03	0.11	0.24
3	16	500	SPSO 2007	48	140	1065.50	114.16	50	0.36	0.04	0.28	0.44
3	16	500	SPSO 2011	48	140	5000.00	0.00	0	0.00	0.00	0.00	0.00

Table 3.4: Summary statistics from the validation study for PSO runs in generating A -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
1	3	50	Basic	3	140	77.00	9.64	140	1.00	0.00	1.00	1.00
1	3	50	SPSO 2007	3	140	89.50	12.60	140	1.00	0.00	1.00	1.00
1	3	50	SPSO 2011	3	140	198.50	37.81	140	1.00	0.00	1.00	1.00
1	3	150	Basic	3	140	69.00	8.90	140	1.00	0.00	1.00	1.00
1	3	150	SPSO 2007	3	140	85.00	14.83	140	1.00	0.00	1.00	1.00
1	3	150	SPSO 2011	3	140	198.00	42.25	140	1.00	0.00	1.00	1.00
1	3	500	Basic	3	140	62.00	7.41	140	1.00	0.00	1.00	1.00
1	3	500	SPSO 2007	3	140	72.00	11.86	140	1.00	0.00	1.00	1.00
1	3	500	SPSO 2011	3	140	208.00	43.74	140	1.00	0.00	1.00	1.00
1	4	50	Basic	4	140	103.00	5.93	140	1.00	0.00	1.00	1.00
1	4	50	SPSO 2007	4	140	131.00	10.38	140	1.00	0.00	1.00	1.00
1	4	50	SPSO 2011	4	140	410.00	105.26	139	0.99	0.01	0.98	1.00
1	4	150	Basic	4	140	94.00	5.93	140	1.00	0.00	1.00	1.00
1	4	150	SPSO 2007	4	140	141.00	20.76	140	1.00	0.00	1.00	1.00
1	4	150	SPSO 2011	4	140	353.00	54.86	140	1.00	0.00	1.00	1.00
1	4	500	Basic	4	140	89.00	4.45	140	1.00	0.00	1.00	1.00
1	4	500	SPSO 2007	4	140	160.00	23.72	140	1.00	0.00	1.00	1.00
1	4	500	SPSO 2011	4	140	354.00	40.77	140	1.00	0.00	1.00	1.00
1	5	50	Basic	5	140	119.00	5.93	140	1.00	0.00	1.00	1.00

Table 3.4: Summary statistics from the validation study for PSO runs in generating *A*-optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
1	5	50	SPSO 2007	5	140	163.00	11.86	140	1.00	0.00	1.00	1.00
1	5	50	SPSO 2011	5	140	1851.50	1103.80	134	0.96	0.02	0.92	0.99
1	5	150	Basic	5	140	110.00	4.45	140	1.00	0.00	1.00	1.00
1	5	150	SPSO 2007	5	140	172.00	18.53	140	1.00	0.00	1.00	1.00
1	5	150	SPSO 2011	5	140	539.00	176.43	140	1.00	0.00	1.00	1.00
1	5	500	Basic	5	140	100.00	4.45	140	1.00	0.00	1.00	1.00
1	5	500	SPSO 2007	5	140	190.00	25.20	140	1.00	0.00	1.00	1.00
1	5	500	SPSO 2011	5	140	477.00	49.67	140	1.00	0.00	1.00	1.00
1	6	50	Basic	6	140	160.00	19.27	140	1.00	0.00	1.00	1.00
1	6	50	SPSO 2007	6	140	209.50	25.95	140	1.00	0.00	1.00	1.00
1	6	50	SPSO 2011	6	140	3841.00	1718.33	24	0.17	0.03	0.11	0.24
1	6	150	Basic	6	140	143.50	17.05	140	1.00	0.00	1.00	1.00
1	6	150	SPSO 2007	6	140	215.50	25.95	140	1.00	0.00	1.00	1.00
1	6	150	SPSO 2011	6	140	3386.50	2392.18	128	0.91	0.02	0.87	0.96
1	6	500	Basic	6	140	128.00	11.86	140	1.00	0.00	1.00	1.00
1	6	500	SPSO 2007	6	140	237.50	35.58	140	1.00	0.00	1.00	1.00
1	6	500	SPSO 2011	6	140	1091.50	319.50	140	1.00	0.00	1.00	1.00
1	7	50	Basic	7	140	118.00	6.67	129	0.92	0.02	0.88	0.97
1	7	50	SPSO 2007	7	140	165.50	16.31	140	1.00	0.00	1.00	1.00
1	7	50	SPSO 2011	7	140	3465.00	1804.32	1	0.01	0.01	0.00	0.02
1	7	150	Basic	7	140	107.00	5.19	139	0.99	0.01	0.98	1.00
1	7	150	SPSO 2007	7	140	173.00	16.31	140	1.00	0.00	1.00	1.00
1	7	150	SPSO 2011	7	140	1450.00	1467.77	12	0.09	0.02	0.04	0.13
1	7	500	Basic	7	140	97.00	4.45	140	1.00	0.00	1.00	1.00
1	7	500	SPSO 2007	7	140	191.00	23.72	140	1.00	0.00	1.00	1.00
1	7	500	SPSO 2011	7	140	849.00	425.51	89	0.64	0.04	0.55	0.72
1	8	50	Basic	8	140	129.00	5.93	140	1.00	0.00	1.00	1.00

Table 3.4: Summary statistics from the validation study for PSO runs in generating *A*-optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
1	8	50	SPSO 2007	8	140	201.00	22.24	140	1.00	0.00	1.00	1.00
1	8	50	SPSO 2011	8	140	3516.00	1357.32	0	0.00	0.00	0.00	0.00
1	8	150	Basic	8	140	117.00	4.45	140	1.00	0.00	1.00	1.00
1	8	150	SPSO 2007	8	140	207.50	18.53	140	1.00	0.00	1.00	1.00
1	8	150	SPSO 2011	8	140	2626.00	2792.48	2	0.01	0.01	0.00	0.03
1	8	500	Basic	8	140	106.00	4.45	140	1.00	0.00	1.00	1.00
1	8	500	SPSO 2007	8	140	235.50	25.95	140	1.00	0.00	1.00	1.00
1	8	500	SPSO 2011	8	140	1270.00	688.67	53	0.38	0.04	0.30	0.46
1	9	50	Basic	9	140	143.00	10.38	127	0.91	0.02	0.86	0.96
1	9	50	SPSO 2007	9	140	240.00	25.20	140	1.00	0.00	1.00	1.00
1	9	50	SPSO 2011	9	140	3724.00	1522.63	0	0.00	0.00	0.00	0.00
1	9	150	Basic	9	140	127.00	7.41	140	1.00	0.00	1.00	1.00
1	9	150	SPSO 2007	9	140	248.00	22.24	140	1.00	0.00	1.00	1.00
1	9	150	SPSO 2011	9	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
1	9	500	Basic	9	140	114.50	3.71	140	1.00	0.00	1.00	1.00
1	9	500	SPSO 2007	9	140	275.00	28.91	140	1.00	0.00	1.00	1.00
1	9	500	SPSO 2011	9	140	1519.50	898.46	17	0.12	0.03	0.07	0.18
2	6	50	Basic	12	140	284.00	48.18	140	1.00	0.00	1.00	1.00
2	6	50	SPSO 2007	12	140	577.00	91.92	140	1.00	0.00	1.00	1.00
2	6	50	SPSO 2011	12	140	2236.00	981.48	0	0.00	0.00	0.00	0.00
2	6	150	Basic	12	140	247.50	46.70	140	1.00	0.00	1.00	1.00
2	6	150	SPSO 2007	12	140	608.00	83.03	140	1.00	0.00	1.00	1.00
2	6	150	SPSO 2011	12	140	3131.00	2022.27	0	0.00	0.00	0.00	0.00
2	6	500	Basic	12	140	227.00	43.00	140	1.00	0.00	1.00	1.00
2	6	500	SPSO 2007	12	140	633.50	74.87	140	1.00	0.00	1.00	1.00
2	6	500	SPSO 2011	12	140	2827.00	3221.69	0	0.00	0.00	0.00	0.00
2	7	50	Basic	14	140	227.00	45.96	58	0.41	0.04	0.33	0.50

Table 3.4: Summary statistics from the validation study for PSO runs in generating *A*-optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
2	7	50	SPSO 2007	14	140	471.50	89.70	71	0.51	0.04	0.42	0.59
2	7	50	SPSO 2011	14	140	2762.00	1099.35	0	0.00	0.00	0.00	0.00
2	7	150	Basic	14	140	197.50	31.88	76	0.54	0.04	0.46	0.63
2	7	150	SPSO 2007	14	140	470.00	84.51	65	0.46	0.04	0.38	0.55
2	7	150	SPSO 2011	14	140	4705.50	436.63	0	0.00	0.00	0.00	0.00
2	7	500	Basic	14	140	173.50	24.46	78	0.56	0.04	0.47	0.64
2	7	500	SPSO 2007	14	140	548.00	105.26	91	0.65	0.04	0.57	0.73
2	7	500	SPSO 2011	14	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
2	8	50	Basic	16	140	228.50	43.74	94	0.67	0.04	0.59	0.75
2	8	50	SPSO 2007	16	140	463.00	77.84	99	0.71	0.04	0.63	0.78
2	8	50	SPSO 2011	16	140	2839.50	1215.73	0	0.00	0.00	0.00	0.00
2	8	150	Basic	16	140	197.50	36.32	101	0.72	0.04	0.65	0.80
2	8	150	SPSO 2007	16	140	492.00	57.82	100	0.71	0.04	0.64	0.79
2	8	150	SPSO 2011	16	140	4218.50	1158.65	0	0.00	0.00	0.00	0.00
2	8	500	Basic	16	140	174.50	28.91	112	0.80	0.03	0.73	0.87
2	8	500	SPSO 2007	16	140	528.50	71.91	114	0.81	0.03	0.75	0.88
2	8	500	SPSO 2011	16	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
2	9	50	Basic	18	140	237.00	54.86	124	0.89	0.03	0.83	0.94
2	9	50	SPSO 2007	18	140	504.00	59.30	139	0.99	0.01	0.98	1.00
2	9	50	SPSO 2011	18	140	2956.00	1321.74	0	0.00	0.00	0.00	0.00
2	9	150	Basic	18	140	193.50	24.46	137	0.98	0.01	0.95	1.00
2	9	150	SPSO 2007	18	140	520.00	49.67	140	1.00	0.00	1.00	1.00
2	9	150	SPSO 2011	18	140	4220.50	1155.69	1	0.01	0.01	0.00	0.02
2	9	500	Basic	18	140	168.50	21.50	138	0.99	0.01	0.97	1.00
2	9	500	SPSO 2007	18	140	568.00	77.10	140	1.00	0.00	1.00	1.00
2	9	500	SPSO 2011	18	140	5000.00	0.00	6	0.04	0.02	0.01	0.08
2	10	50	Basic	20	140	234.00	41.51	101	0.72	0.04	0.65	0.80

Table 3.4: Summary statistics from the validation study for PSO runs in generating *A*-optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
2	10	50	SPSO 2007	20	140	533.50	52.63	124	0.89	0.03	0.83	0.94
2	10	50	SPSO 2011	20	140	3337.50	1699.06	0	0.00	0.00	0.00	0.00
2	10	150	Basic	20	140	181.50	22.98	113	0.81	0.03	0.74	0.87
2	10	150	SPSO 2007	20	140	540.00	48.93	136	0.97	0.01	0.94	1.00
2	10	150	SPSO 2011	20	140	3843.00	1715.37	0	0.00	0.00	0.00	0.00
2	10	500	Basic	20	140	155.50	18.53	129	0.92	0.02	0.88	0.97
2	10	500	SPSO 2007	20	140	558.00	35.58	140	1.00	0.00	1.00	1.00
2	10	500	SPSO 2011	20	140	4620.00	563.39	0	0.00	0.00	0.00	0.00
2	11	50	Basic	22	140	261.00	49.67	52	0.37	0.04	0.29	0.45
2	11	50	SPSO 2007	22	140	586.00	61.53	91	0.65	0.04	0.57	0.73
2	11	50	SPSO 2011	22	140	3051.50	1401.80	0	0.00	0.00	0.00	0.00
2	11	150	Basic	22	140	204.00	23.72	75	0.54	0.04	0.45	0.62
2	11	150	SPSO 2007	22	140	594.50	55.60	118	0.84	0.03	0.78	0.90
2	11	150	SPSO 2011	22	140	3595.00	1722.78	0	0.00	0.00	0.00	0.00
2	11	500	Basic	22	140	176.00	26.69	100	0.71	0.04	0.64	0.79
2	11	500	SPSO 2007	22	140	644.00	49.67	140	1.00	0.00	1.00	1.00
2	11	500	SPSO 2011	22	140	4378.50	921.44	0	0.00	0.00	0.00	0.00
2	12	50	Basic	24	140	294.50	83.77	79	0.56	0.04	0.48	0.65
2	12	50	SPSO 2007	24	140	664.00	88.96	113	0.81	0.03	0.74	0.87
2	12	50	SPSO 2011	24	140	3357.00	1976.31	0	0.00	0.00	0.00	0.00
2	12	150	Basic	24	140	236.50	34.84	101	0.72	0.04	0.65	0.80
2	12	150	SPSO 2007	24	140	699.00	63.75	133	0.95	0.02	0.91	0.99
2	12	150	SPSO 2011	24	140	4318.50	1010.39	0	0.00	0.00	0.00	0.00
2	12	500	Basic	24	140	200.00	28.91	130	0.93	0.02	0.89	0.97
2	12	500	SPSO 2007	24	140	742.00	58.56	140	1.00	0.00	1.00	1.00
2	12	500	SPSO 2011	24	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	10	50	Basic	30	140	724.00	395.11	51	0.36	0.04	0.28	0.45

Table 3.4: Summary statistics from the validation study for PSO runs in generating *A*-optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
3	10	50	SPSO 2007	30	140	941.50	189.77	76	0.54	0.04	0.46	0.63
3	10	50	SPSO 2011	30	140	4280.50	1066.73	0	0.00	0.00	0.00	0.00
3	10	150	Basic	30	140	336.50	82.28	64	0.46	0.04	0.37	0.54
3	10	150	SPSO 2007	30	140	985.00	229.06	64	0.46	0.04	0.37	0.54
3	10	150	SPSO 2011	30	140	4603.00	588.59	0	0.00	0.00	0.00	0.00
3	10	500	Basic	30	140	263.00	57.08	56	0.40	0.04	0.32	0.48
3	10	500	SPSO 2007	30	140	993.00	189.03	87	0.62	0.04	0.54	0.70
3	10	500	SPSO 2011	30	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	11	50	Basic	33	140	952.00	369.91	42	0.30	0.04	0.22	0.38
3	11	50	SPSO 2007	33	140	1004.00	131.21	87	0.62	0.04	0.54	0.70
3	11	50	SPSO 2011	33	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	11	150	Basic	33	140	385.00	126.76	71	0.51	0.04	0.42	0.59
3	11	150	SPSO 2007	33	140	971.00	115.64	97	0.69	0.04	0.61	0.77
3	11	150	SPSO 2011	33	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	11	500	Basic	33	140	281.50	60.79	79	0.56	0.04	0.48	0.65
3	11	500	SPSO 2007	33	140	979.50	85.25	112	0.80	0.03	0.73	0.87
3	11	500	SPSO 2011	33	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	12	50	Basic	36	140	1121.00	322.47	6	0.04	0.02	0.01	0.08
3	12	50	SPSO 2007	36	140	1087.50	199.41	48	0.34	0.04	0.26	0.42
3	12	50	SPSO 2011	36	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	12	150	Basic	36	140	466.50	222.39	34	0.24	0.04	0.17	0.32
3	12	150	SPSO 2007	36	140	1116.50	250.56	46	0.33	0.04	0.25	0.41
3	12	150	SPSO 2011	36	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	12	500	Basic	36	140	294.50	61.53	37	0.26	0.04	0.19	0.34
3	12	500	SPSO 2007	36	140	1143.00	254.27	55	0.39	0.04	0.31	0.48
3	12	500	SPSO 2011	36	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	13	50	Basic	39	140	1104.00	305.42	4	0.03	0.01	0.00	0.06

Table 3.4: Summary statistics from the validation study for PSO runs in generating *A*-optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
3	13	50	SPSO 2007	39	140	1278.00	317.28	12	0.09	0.02	0.04	0.13
3	13	50	SPSO 2011	39	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	13	150	Basic	39	140	666.50	399.56	7	0.05	0.02	0.01	0.09
3	13	150	SPSO 2007	39	140	1294.50	269.83	13	0.09	0.02	0.04	0.14
3	13	150	SPSO 2011	39	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	13	500	Basic	39	140	348.00	93.40	10	0.07	0.02	0.03	0.11
3	13	500	SPSO 2007	39	140	1457.00	297.26	16	0.11	0.03	0.06	0.17
3	13	500	SPSO 2011	39	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	14	50	Basic	42	140	1141.50	254.27	3	0.02	0.01	0.00	0.05
3	14	50	SPSO 2007	42	140	1188.00	165.31	2	0.01	0.01	0.00	0.03
3	14	50	SPSO 2011	42	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	14	150	Basic	42	140	703.50	361.01	3	0.02	0.01	0.00	0.05
3	14	150	SPSO 2007	42	140	1231.00	164.57	0	0.00	0.00	0.00	0.00
3	14	150	SPSO 2011	42	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	14	500	Basic	42	140	354.00	114.90	3	0.02	0.01	0.00	0.05
3	14	500	SPSO 2007	42	140	1271.00	198.67	4	0.03	0.01	0.00	0.06
3	14	500	SPSO 2011	42	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	15	50	Basic	45	140	1111.00	257.97	0	0.00	0.00	0.00	0.00
3	15	50	SPSO 2007	45	140	1247.00	181.62	4	0.03	0.01	0.00	0.06
3	15	50	SPSO 2011	45	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	15	150	Basic	45	140	810.00	378.06	2	0.01	0.01	0.00	0.03
3	15	150	SPSO 2007	45	140	1263.00	120.83	2	0.01	0.01	0.00	0.03
3	15	150	SPSO 2011	45	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	15	500	Basic	45	140	393.50	137.14	6	0.04	0.02	0.01	0.08
3	15	500	SPSO 2007	45	140	1325.50	168.28	6	0.04	0.02	0.01	0.08
3	15	500	SPSO 2011	45	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	16	50	Basic	48	140	1215.50	234.25	1	0.01	0.01	0.00	0.02

Table 3.4: Summary statistics from the validation study for PSO runs in generating A -optimal designs

K	N	S	psover	dimension	n.run	med.iter	mad.iter	n.sucss	pr.sucss	se.sucss	lb	ub
3	16	50	SPSO 2007	48	140	1269.00	138.62	2	0.01	0.01	0.00	0.03
3	16	50	SPSO 2011	48	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	16	150	Basic	48	140	991.50	416.61	2	0.01	0.01	0.00	0.03
3	16	150	SPSO 2007	48	140	1348.00	166.79	3	0.02	0.01	0.00	0.05
3	16	150	SPSO 2011	48	140	5000.00	0.00	0	0.00	0.00	0.00	0.00
3	16	500	Basic	48	140	457.50	172.72	2	0.01	0.01	0.00	0.03
3	16	500	SPSO 2007	48	140	1397.00	163.83	7	0.05	0.02	0.01	0.09
3	16	500	SPSO 2011	48	140	5000.00	0.00	0	0.00	0.00	0.00	0.00

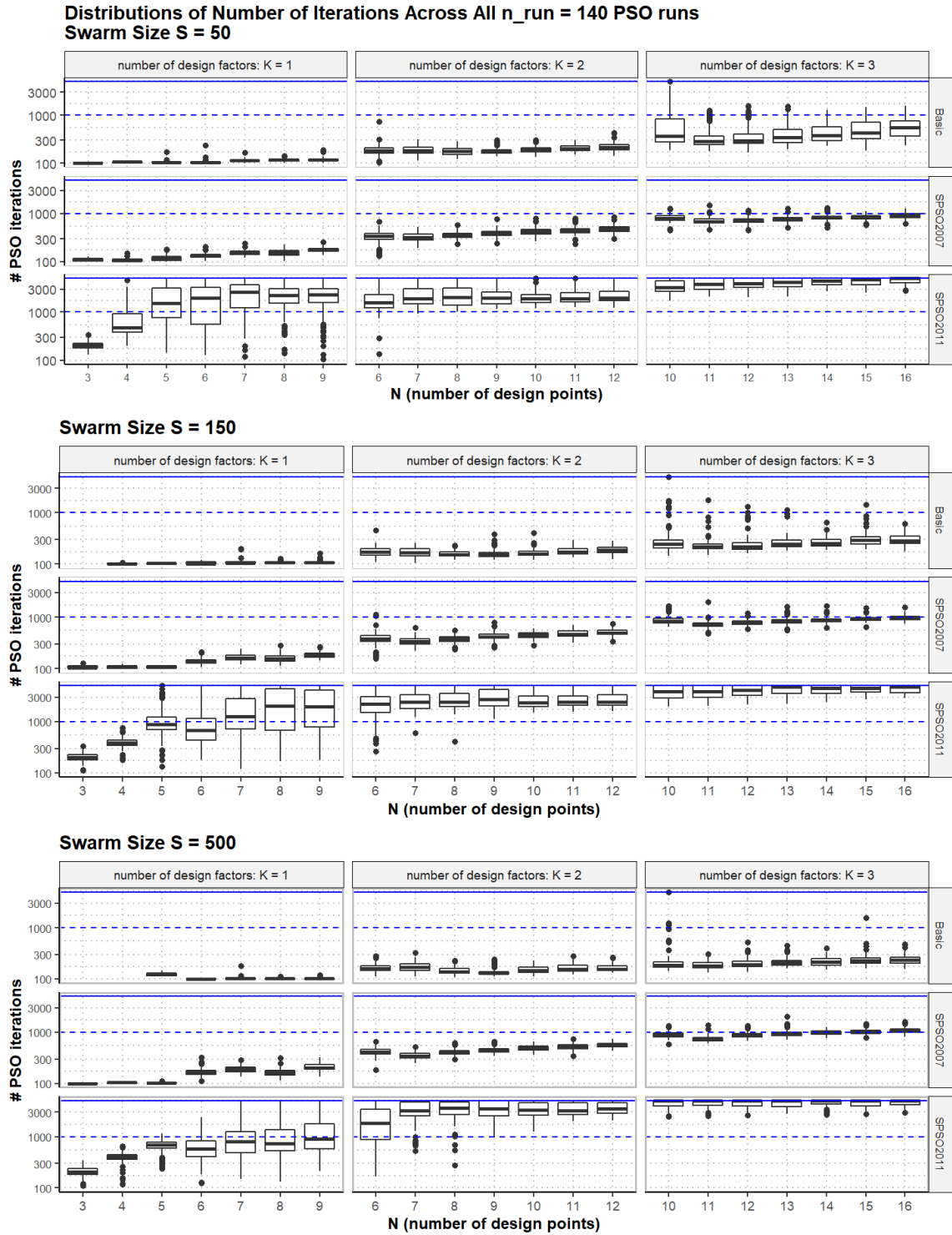


Figure 3.17: *D*-optimal design: Boxplots of number of iterations to termination of each PSO algorithm across the study space. The dotted blue line is at 1000 iterations while the solid blue line is at 5000 iterations.

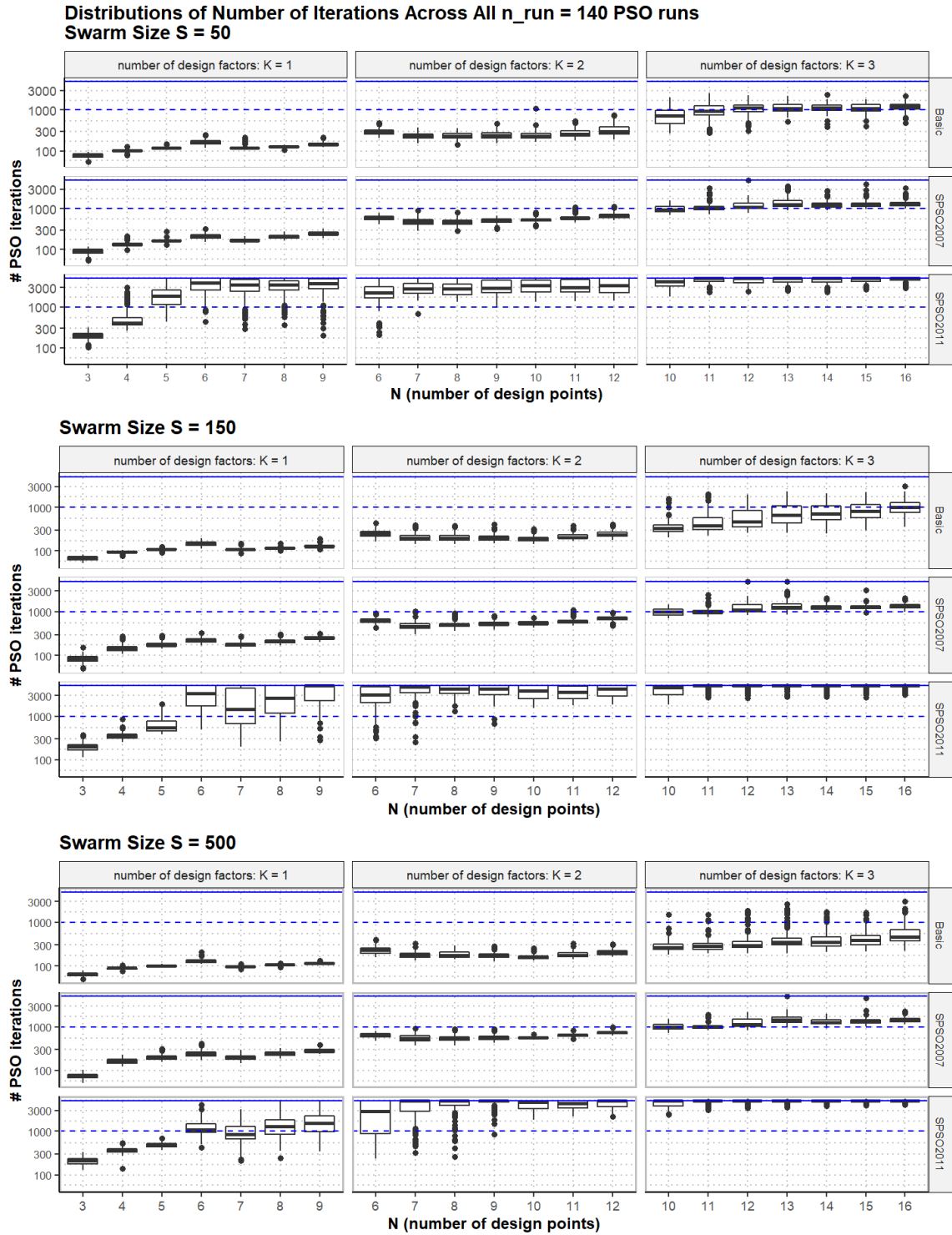


Figure 3.18: *A*-optimal design: Boxplots of number of iterations to termination of each PSO algorithm across the study space. The dotted blue line is at 1000 iterations while the solid blue line is at 5000 iterations.

CHAPTER FOUR

FAST COMPUTATION OF HIGHLY EFFICIENT EXACT G -OPTIMAL
DESIGNS VIA PARTICLE SWARM OPTIMIZATION AND PARALLEL
COMPUTINGABSTRACT

In this chapter we summarize the state-of-the-art (SOA) for generating G -optimal exact response surface designs on the hypercube. We discuss three main papers that address the difficulties of computing G -optimal designs, provide advancement of algorithms to generate G -optimal designs, and publish the best known G -optimal designs found to date. We demonstrate that PSO is competitive to the SOA algorithms both in computing speed and ability to generate G -optimal designs with high efficiency.

4.1 Introduction

The first paper on generating G -optimal designs, which, incidentally, is the seminal paper for the entire field of optimal design of experiments, was published in *Biometrika* in 1918 by Kristine Smith [87]. Smith, a Danish statistician, worked at Karl Pearson's laboratory. Another portion of her work, concerning the use of the Chi-squared statistic, was criticized by Sir Ronald Fisher. His attempts to publish his criticisms were twice rejected by Pearson and fueled a feud between these two fathers of modern statistics [5]. Nonetheless, her 1918 paper is now recognized as the genesis of the theory of optimal design. In her paper, she provided exposition on how the standard deviation, as a function of the sample/design points, instructs the optimal placement of samples for one-factor models used to fit polynomials up to degree 6.

Atkinson and Bailey state that her paper on these designs was at least 30 years before its time, and are those designs that are generated by the theory of Kiefer [5, 58].

The literature published in the last two decades for generating exact G -optimal designs contains three primary papers. The first paper is that of Borkowski (2003) [13]. In this paper the author adapted genetic algorithms (GA) to the purpose of generating small exact response surface designs on the hypercube for the D , A , IV , and G criteria. This highly cited paper produced the best known exact optimal designs published to that point for the second-order model. The results presented in this paper have become a standard research benchmark for which to compare the performance of new algorithms and designs. The author covered $K = 1, 2, 3$ factor models with seven design point (N) scenarios. Both the most efficient designs for each criterion and their criterion scores are published.

The second paper on the topic was published by Rodriguez et.al. (2010) [78]. This paper contains a new algorithm for generating near G -optimal designs for second-order models on the hypercube. The algorithm is an application of Brent's minimization algorithm combined with coordinate exchange. We note that starting seeds for their algorithm include the D and I -optimal designs, some random designs, and some user specified designs. The authors covered all $K = 1, 2, 3$ factor designs as presented in Borkowski (2003), and extended the searches to three scenarios on each of $K = 4, 5$ factor models. This extension demonstrated an advancement to computing capability because G -optimal design-generating algorithms are computationally expensive (this will be discussed in subsequent sections). The authors replicated the G -optimal designs for $K = 1, 2, 3$ factors as published in Borkowski (2003) and were able to demonstrate that the $K = 4, 5$ factor models they generated had higher G -optimality than did published designs on the same number of factors and design points. The authors also did a detailed comparison of the G -optimal designs to

the IV -optimal designs via fraction of design space (FDS) plots which illustrated that while the G -optimal models had lower maximum prediction variance, they had larger prediction variances as compared to IV -optimal designs over a large fraction of the hypercube (thus, generating a general recommendation that practitioners should prefer IV -optimal models in application).

The third paper is a recent publication by Hernandez and Nachtsheim (2018) [47] which proposes a fast computation of G -optimal Designs via exploiting a new-found relationship with the I_λ optimality criterion. I_λ is an extension to the IV -criterion, on continuous designs, (i.e., average prediction variance computed via an integration over the design space) where λ represents an importance weighting on the unique points of the continuous design. The authors are able to use fast algorithms to construct continuous I_λ -optimal designs and in turn use these as approximations to exact G -optimal designs. A large portion of the the author's work essentially focuses on an efficient way to find values to populate the weight matrix

$$\mathbf{W} := \int_{\mathcal{X}} \mathbf{f}'(\mathbf{x}')\mathbf{f}(\mathbf{x}')\lambda(\mathbf{x}')d\mathbf{x}'. \quad (4.1)$$

They start the algorithm by appealing to the general equivalence theorem for approximate designs [6]. They claim that starting with a $G(D)$ -optimal design (i.e., is the continuous optimal design via the general equivalence theorem) will initiate the search problem for $\mathbf{W}$ in the neighborhood of the true solution (steps 1. and 2. in algorithm on page 10 of [47]). They build a local grid around the $G(D)$ -optimal defined $\mathbf{W}$, and then use the coordinate exchange algorithm on I_λ optimality to refine the entries of $\mathbf{W}$. From the identified exact I_λ optimal design, they compute the maximum prediction variance for that design, using a grid based approach similar to [13, 78] and use it as an approximation of the true G -criterion. They iterate and

retain the design with the maximum G -criterion found during the search. To illustrate the validity of this new algorithm for generating G -optimal designs they compare to two different algorithms which have produced the most current knowledge on G -optimal designs to date: (1.) the state-of-practice algorithm G -CEXCH (a coordinate exchange algorithm) as implemented in the scripting language of the JMP commercial statistical software developed by Rodriguez et. al. (2010) [78], and (2.) results from the genetic algorithm (GA) approach to generating G -optimal designs as implemented by Borkowski (2003) [13]. The results can be briefly summarized as follows:

1. In Table 1 of [47] the authors compare the G -optimal designs generated via their approach for all design scenarios considered in Borkowski (2003) and compare their designs, via efficiency, to designs generated by G -CEXCH and GA. The general story is that their designs are often but not always as efficient as G -CEXCH designs, and further, in all but one case, their designs were equally or less efficient as GA designs. These results suggests that the designs published by Borkowski (2003) [13] remain state-of-the art and the ‘benchmark to beat’ to date.
2. The strength of the author’s algorithm is in computational speed. In terms of number of function evaluations their algorithm is approximately as expensive as G -CEXCH, but about 1 to 2 orders of magnitude cheaper than GA. With respect to computing time, their algorithm is more expensive than G -CEXCH for the $K = 1$ low dimensional design problems, but many times cheaper (3 to 18 times) for the $K = 2, 3$ factor design problems. With respect to GA, their algorithm is many factors cheaper in computing time (ranging from 3 to 100 times cheaper across $K = 1, 2, 3$ factor design scenarios).
3. They pushed their algorithm into higher dimensional design problems with

$K = 4, 5$ factors. These spaces are often neglected for the G -optimal design problem due to the computational burden involved. They only implemented the algorithm on $N = 17$ and 23 runs respectively. They state that the computing time for either GA or G-CEXCH prohibit their application on these problems (e.g., they state the G-CEXCH would take 25 and 166 days, respectively, to search for these designs). They compare their G -optimal designs for these scenarios to the D -optimal designs via fraction of design space plots.

In subsequent sections we will discuss the computational hurdles for quick generation of exact G -optimal designs and will demonstrate that PSO is superior to current algorithms in both computing speed and its ability to find highly efficient G -optimal designs.

4.2 Scoring Candidate Designs on the G -optimality Scale.

Exact G -optimal models are rarely applied in practical applications. One of the reasons for this is that the search for exact G -optimal designs requires that each candidate design is scored via an optimization (i.e., the maximum prediction variance over $\mathcal{X}$ must be found for the candidate design $\mathbf{X}$). Formally, the G -score for a candidate design is

$$G(\mathbf{X}) = \max_{\mathbf{x}' \in \mathcal{X}} N \mathbf{f}'(\mathbf{x}') (\mathbf{F}' \mathbf{F})^{-1} \mathbf{f}(\mathbf{x}') \quad (4.2)$$

and the G -optimal exact design is defined as

$$\begin{aligned} \mathbf{X}^* &:= \operatorname{argmin}_{\mathbf{X} \in \mathcal{X}^N} G(\mathbf{X}) \\ &= \operatorname{argmin}_{\mathbf{X} \in \mathcal{X}^N} \max_{\mathbf{x}' \in \mathcal{X}} N \mathbf{f}'(\mathbf{x}') (\mathbf{F}' \mathbf{F})^{-1} \mathbf{f}(\mathbf{x}'). \end{aligned} \quad (4.3)$$

Eq. 4.3 illustrates that finding the G -optimal design is a minimax problem, such that for every candidate $\mathbf{X}$, an accurate way of computing the G -score in Eq. 4.2 is required.

The primary computational issue is that, while $\mathcal{X}$ is a convex set (i.e., its the hypercube), $G(\mathbf{X})$ is not a convex function due to the mapping of the design point $\mathbf{x}'$ to the model vector $\mathbf{f}'(\mathbf{x}')$. The G -surface is multimodal and it is difficult to predict the nature of the surface for an arbitrary candidate design. Thus, the problem of scoring a candidate design on the G -scale is mathematically intractable and standard gradient based algorithms cannot be applied without serious adjustment due to susceptibility to becoming entrapped at local optima. To give some feeling for the nature of the G -surface, we present a set of contour plots of the G -score of the G , D , I optimal designs, as well as a random design for the $K = 2$ $N = 11$ in Figure 4.1. This figure illustrates that in all cases scoring a matrix via the G -criterion is a multimodal optimization (this is because G is a fourth-order polynomial function of the factors). This fact has, and still, remains a challenge for generating exact optimal design. That is, to make the search for G -optimal designs valid, it is necessary that every candidate matrix is accurately scored so that false signals don't contaminate the search.

To assign a G -score to a candidate matrix, Rodriguez et al. apply Brent's univariate optimization algorithm. The algorithm is started at a random point in the design region and cycles through the coordinates until no increase in prediction variance is found. Because there is no guarantee that the algorithm will not get trapped at a local optimum when scoring a candidate design, the algorithm is repeated 100 times for each candidate design. The authors validated this approach by applying the algorithm to score a set of designs with known G -score [78]. In contrast, both Borkowski (2003) [13, 16] and Hernandez and Nachtsheim (2018) [47] found that a

course grid over $\mathcal{X}$ worked sufficiently well to approximate $G(\mathbf{X})$ for a given design. Specifically, a 5^K grid with $x_j \in \{-1, -0.5, 0, 0.5, 1\}$ for $j = 1, 2, \dots, K$ has performed well for approximating $G(\mathbf{X})$.

In light of these observations (and due to the complexity of the calculation involved), we follow the approach of Borkowski (2003) and Hernandez and Nachtsheim (2018) and also employ the 5^K grid to score candidate matrices on the G -scale during PSO searches. This will ensure that our results are directly comparable to the previous published results.

While we have posed PSO searches as a minimization calculation, we will report the results on the scale provided by Borkowski (2003), specifically designs will be scored as

$$G_{\text{eff}}(\mathbf{X}) = \frac{100p}{\max_{\mathbf{x}' \in \mathcal{X}} N\mathbf{f}'(\mathbf{x}')(\mathbf{F}'\mathbf{F})^{-1}\mathbf{f}(\mathbf{x}')}. \quad (4.4)$$

On this scale, larger G implies a design with higher G -optimality and the maximum value on this scale is 100. We will compare the PSO generated designs to the GA generated designs of Borkowski (2003) using

$$G_{\text{releff}}(\mathbf{X}_{\text{PSO}}, \mathbf{X}_{\text{GA}}) = 100 \frac{G(\mathbf{X}_{\text{PSO}})}{G(\mathbf{X}_{\text{GA}})} \quad (4.5)$$

where $\mathbf{X}_{\text{GA}}$ is the optimal design generated by the GA [13].

4.3 Small Exact PSO-Generated G -optimal Response Surface Designs

In Chapter 2 we demonstrated that SPSO2007 (with local communication topology) performed better than Basic PSO and SPSO2011 with respect to the probability of generating the D and A -optimal designs. Thus, in this section we only employ SPSO2007 to search for G optimal designs. We cover all 21 design scenarios as in Borkowski (2003) [13], specifically:

1. number of Design Factors - $K \in \{1, 2, 3\}$, and
2. number of Design Points - 7 levels per K :

$$K = 1 \implies N \in \{3, 4, 5, 6, 7, 8, 9\},$$

$$K = 2 \implies N \in \{6, 7, 8, 9, 10, 11, 12\},$$

$$K = 3 \implies N \in \{10, 11, 12, 13, 14, 15, 16\},$$

We used a swarm size of $S = 150$ particles and ran $n_{\text{run}} = 140$ independent PSO searches (run in parallel across 7 CPU threads). The PSO algorithm was allowed to run until a non-zero difference between previous and current best fitness is less than the square root of machine epsilon, specifically $|f_{t-1}(\mathbf{G}_{\text{best}}) - f_t(\mathbf{G}_{\text{best}})| < 1.49E - 8$. Total run time for this set of searches was approximately 30 minutes with the $K = 1$ searches taking less than a minute for all 140 runs and the $K = 3$, $N = 16$ set of 140 runs taking approximately 4 minutes.

4.3.1 Results Summary

In Figure 4.2 we present boxplots of the G -scores of the PSO generated designs for each scenario (top panel) and the efficiencies of the PSO designs relative to the GA designs (bottom panel). These graphics illustrate that in all cases PSO found the GA optimal designs, and, in the higher dimensional cases PSO has identified designs that have higher G -efficiency than the GA designs. This is evidenced by the existence of points in the boxplots showing over 100% efficiency.

4.3.2 Comparison to State-of-the-Art Algorithms

Hernandez and Nachtsheim (2018) provide a comprehensive summary of the performance of their algorithm, which we will refer to as $G(I_\lambda)$ -CEXCH, to the G -CEXCH algorithm presented in Rodriguez et.al. (2010) and to the performance of

the GA as applied in Borkowski (2003) [13, 47, 78]. In their study, they ran each algorithm on all 21 design scenarios using 200 random starts (they did not use the results as published in either paper). Their intent in doing this was also to study the computational burden (i.e., computing time) on modern machines in order to demonstrate that $G(I_\lambda)$ -CEXCH is relatively fast at generating G -efficient designs. We will use the information published in Table 1 of Hernandez and Nachtsheim (2018), wherein the authors provide the efficiencies of the best designs found by $G(I_\lambda)$ -CEXCH relative to the best designs found by each of the G -CEXCH and GA algorithms. In order to give a direct comparison of the performance of all 4 algorithms, $G(I_\lambda)$ -CEXCH, G -CEXCH, GA, and PSO (which we refer to as G -PSO), we will report efficiencies of the best designs found by $G(I_\lambda)$ -CEXCH, G -CEXCH, and G -PSO relative to the best design found by GA as reported in Borkowski (2003). Because Hernandez and Nachtsheim (2018) ran the G -CEXCH and GA algorithms themselves, their GA designs are close but not identical to those of Borkowski (2003). Fortunately, Hernandez and Nachtsheim (2018) reported the best designs generated by the $G(I_\lambda)$ -CEXCH algorithm as supplementary material, and so we were able to compute the G -scores for each design. Using the resulting G -scores, the efficiencies can be computed using the G -scores provided in Borkowski (2003). This also enabled conversion of the $G(I_\lambda)$ -CEXCH efficiencies relative to G -CEXCH reported in Hernandez and Nachtsheim (2018) to G -CEXCH efficiencies relative to the GA designs reported in Borkowski (2003). The resulting efficiencies of designs from all three algorithms relative to GA designs are reported in Table 4.1. The table shows that in all cases the G -PSO designs are as or more efficient than GA, $G(I_\lambda)$ -CEXCH, and G -CEXCH designs. This is encouraging because, in contrast to the 200 random starts used to generate the $G(I_\lambda)$ -CEXCH, and G -CEXCH designs, we used only 140 random starts to generate the G -PSO designs.

Regarding computing time (wallclock) we cannot compare PSO run times to those of Hernandez and Nachtsheim (2018) because they reported runtimes of G -CEXCH and GA relative to the runtime of $G(I_\lambda)$ -CEXCH. We report the number of function evaluations by G -PSO using $S = 150$ particles in the $n_{\text{run}} = 140$ runs in Table 4.2. These can be compared to run times of the $n_{\text{run}} = 200$ random starts of each algorithm shown in Table 2 in Hernandez and Nachtsheim (2018) (their paper did not include the base of the logarithm but we confirmed via email correspondence on June 20, 2020 that they also used log base 10). The number of function evaluations required by G -PSO are on the same order of $G(I_\lambda)$ -CEXCH and G -CEXCH and fewer than GA. This in conjunction with the results in Table 4.1 strongly support the superiority of G -PSO to cheaply generate exact G -optimal designs with a high probability of returning highest efficiency designs.

4.3.3 New Small Exact G -optimal Response Surface Designs

The new two-factor G -optimal PSO designs are reported in Table 4.3. The new three-factor G -optimal PSO designs are reported in Tables 4.4 and 4.5. The G -efficiency for PSO and GA designs are reported as well as the G -releff relative to the best known GA designs.

4.4 Extending G -optimal design generating algorithms to $K=4,5$ factors

It has proven difficult to extend generating algorithms for G -optimal designs to models with more than $K = 3$ factors due to the min/max problem shown in Eq. 4.2. Hernandez and Nachtsheim (2018) [47] discuss this in detail and provide a timing comparison of the GA, G -CEXCH and $G(I_\lambda)$ -CEXCH for the $K = 4$, $N = 17$ and $K = 5$, $N = 23$ design scenarios. They indicated the following:

- For $K = 4$, $N = 17$ and 200 random starts of $G(I_\lambda)$ -CEXCH, it took 20.15h

to finish the searches on their machine. When attempting to apply G -CEXCH they found it took 3h to complete a single random start, and so estimated it would take 25 computing days to run 200 random starts with this algorithm and so did not run it. GA was not feasible for these searches as it already took days and even weeks to complete the $K = 1, 2, 3$ factor searches.

- For $K = 5$, $N = 23$ and 200 random starts of $G(I_\lambda)$ -CEXCH, it took 25.07h to finish the searches on their machine. When attempting to apply G -CEXCH they found it took 20h to complete a single random start, and so estimated it would take 166 computing days to run 200 random starts with this algorithm and so did not run it. GA was not feasible for these searches as it already took days and even weeks to complete the $K = 1, 2, 3$ factor searches.

We note that these authors employed the same 5^K grid as Borkowski (2003) to evaluate Eq. 4.3 and thus score a design on the G -optimality scale.

Rodriguez et.al. (2010) publish results for $K = 4$, $N = 15, 20, 24$ and $K = 5$, $N = 21, 26, 30$ design scenarios. They make no mention of computing time, and, in contrast to Borkowski (2003) and Hernandez and Nachtsheim (2018) they do not use the 5^K grid to store a candidate design. In contrast they apply Brent's minimization algorithm with multiple starts for each candidate design in an effort to find the maximum prediction variance for each candidate design.

Neither of the two papers published G -criterion values for their designs. They did, however, publish their designs allowing us to compute these values for a direct comparison of the quality of G -PSO generated designs for these scenarios to their results. To do this, we will score all designs using the 5^K grid approach.

In scoring the designs from the two papers, some errors were detected. In scoring the $K = 5$ $N = 21, 26$ designs our G -efficiencies did not correspond to the published

results. We reached out to both Myrta Rodriguez and Bradley Jones via email. Both authors kindly responded. Dr. Jones provided a technical response indicating that their published results were mis-scored on the G -efficiency scale due to a drawback of their method and confirmed true scores which were consistent with our scoring (personal correspondence June 8, 2020). The $K = 5$ factor $N = 23$ point design of Hernandez and Nachtsheim (2018) was only published in the supplementary materials with $N = 22$ points. The authors confirmed by email on June 20, 2020 that the missing point was another vertex point.

4.4.1 Comparison of $K=4,5$ factor G -PSO designs to published G -optimal designs

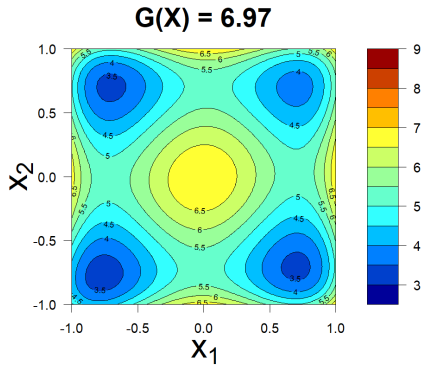
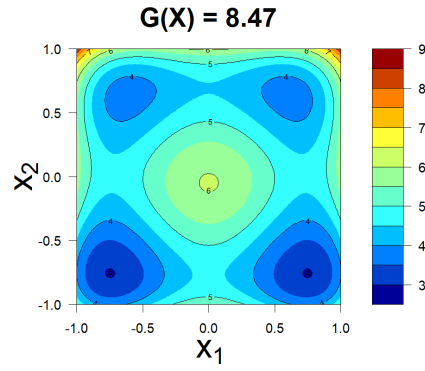
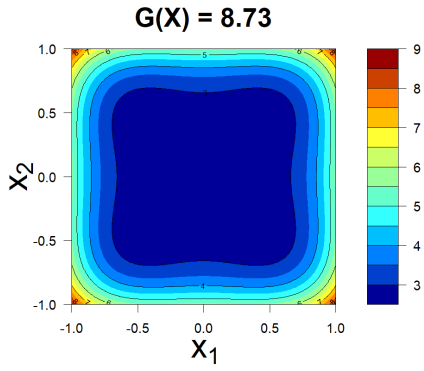
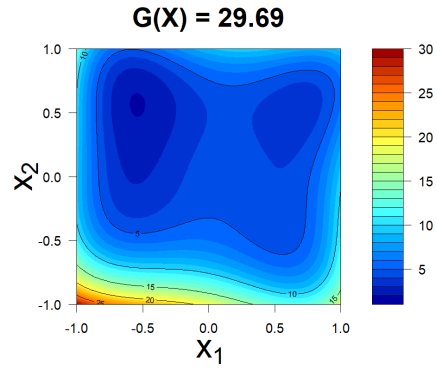
We ran $n_{\text{run}} = 210$ applications of PSO on each of the eight $K = 4, 5$ design scenarios using $S = 150$ particles. Note that these runs were parallelized over 7 CPU cores. The $K = 4$, $N = 15, 17, 20, 24$ scenarios took 37, 35, 49 and 55 minutes to run, respectively (compared to the 20.15h that the $N = 17$ search took for 200 runs of the $G(I_\lambda)$ -CEXCH algorithm). The $K = 5$, $N = 21, 23, 26, 30$ runs of G -PSO took 9h42m, 9h8m, 8h10m, and 9h25m, respectively (compare to the 25.07h that the $N = 23$ scenario took for the 200 runs of the $G(I_\lambda)$ -CEXCH algorithm). The large jump in runtime for G -PSO between the $K = 4$ and $K = 5$ factor cases is related to the gridsize used to score each candidate design during the search. The number of points in the grid for $K = 4$ is $5^4 = 625$ while for $K = 5$ it is $5^5 = 3125$.

We present efficiencies of the best G -PSO designs as well as the best published designs generated for CEXCH approaches in Table 4.6. In all cases G -PSO generated as or more efficient designs than the coordinate exchange approaches. We present boxplots of the G -PSO efficiencies from the 210 runs, as relative efficiencies to published designs, in Figure 4.3. The goal is for the maximum value of the boxplot to be as large or larger than the best existing G -criterion value.

The G -PSO designs for $K = 4$ $N = 15, 17, 20, 24$ are contained in Tables 4.7, 4.8, 4.9, 4.10, respectively. The G -PSO designs for $K = 5$ $N = 21, 23, 26, 30$ are contained in Tables 4.11, 4.12, 4.13, 4.14, respectively.

4.5 Conclusions

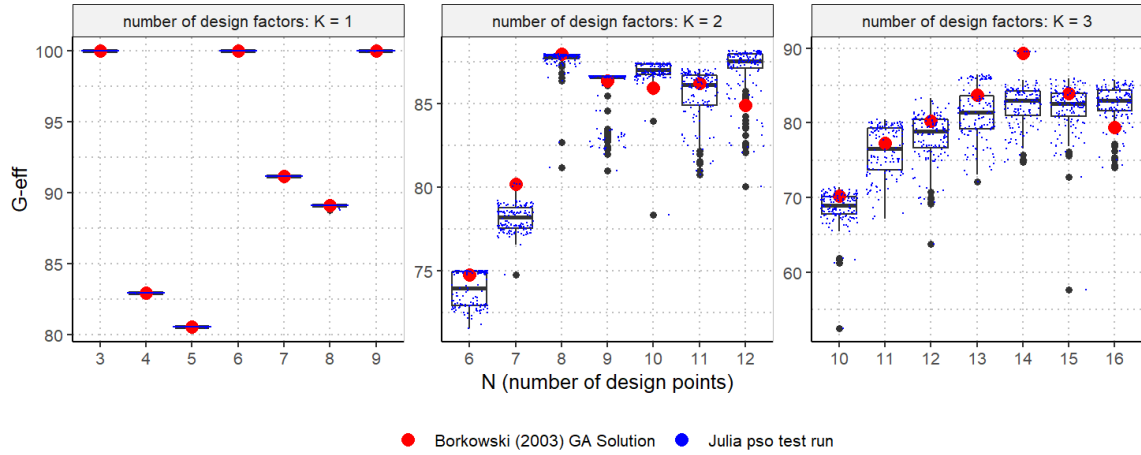
Generating exact G -optimal designs is a challenging problem. There are only three papers in the last 17 years that address this problem. Design searches for more than $K = 3$ factors are especially difficult and costly. In total, the results of this chapter speak extremely favorably to PSO as a generating algorithm for these designs. We have demonstrated that PSO can generate designs which are more efficient than currently published results, and for $K \leq 4$ factors can do this much cheaper than coordinate exchange algorithms (with respect to run time). Although there can be some approximation error to scoring a candidate design on the G -scale using the 5^K grid approach, the designs published herein are highly G -efficient, and there may yet exist slightly better designs. We did compute the G efficiencies for all optimal $K = 1, 2, 3$ factor designs over a finer 11^K grid and found that only a few had G -score errors larger than 1%. This indicates that the 5^K grid approach remains a reasonable way of evaluating the G -efficiency of a candidate design. In future work we plan to replace the inner optimization of the G -criterion with a PSO algorithm.

(a) G -optimal design(b) D -optimal design(c) IV -optimal design

(d) Random design

Figure 4.1: SPV surfaces of the second-order model for the G , D , IV , and random designs for the $K = 2$, $N = 11$ design scenario. The scaling of the z -axis is the same for sub-figures (a), (b), and (c) and so colors can be directly compared in these figures. The G -score of a design is the maximum SPV over $\mathcal{X}$ (i.e., the plot square).

Distributions of G-efficiency of Best Designs found by PSO
Swarm Size $S = 150$



Distributions of G-releff of Best Designs found by PSO
Swarm Size $S = 150$ vs. GA designs

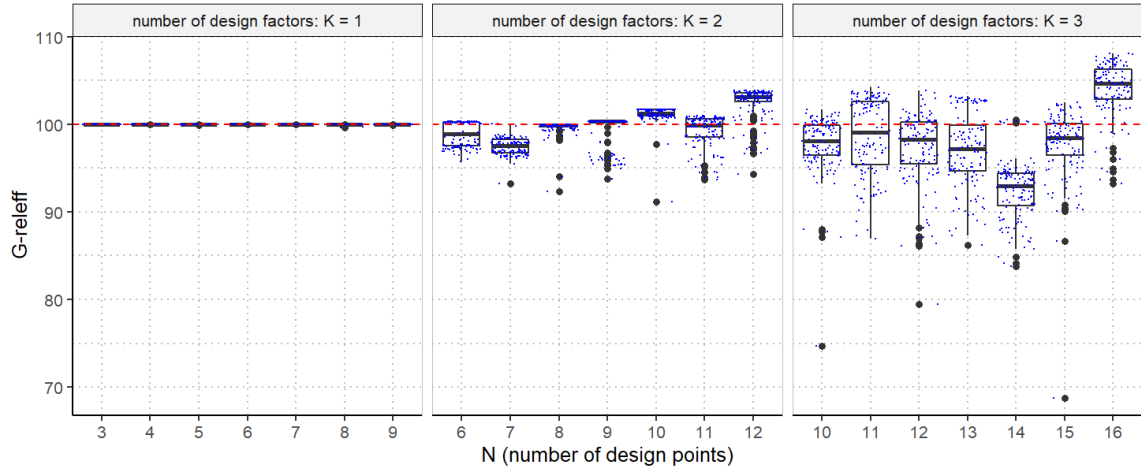


Figure 4.2: The top set of panels shows boxplots of the $n_{\text{run}} = 140$ G -efficiencies of the designs generated by PSO. The red dot represents the G -eff of the GA generated design from Borkowski (2003). The bottom set of panels show boxplots of the G -relative efficiency of the PSO generated designs vs. the GA designs.

Table 4.1: Relative efficiencies of G -optimal designs for each algorithm relative to GA generated designs.

$K = 1$ Factor			
N	G -CEXCH	$G(I_\lambda)$ -CEXCH	G -PSO
3	100.0	100.0	100.0
4	98.7	96.2	100.0
5	98.7	97.0	100.0
6	100.0	100.0	100.0
7	99.7	98.8	100.0
8	99.4	94.7	100.0
9	89.4	100.0	100.0

$K = 2$ Factors			
N	G -CEXCH	$G(I_\lambda)$ -CEXCH	G -PSO
6	96.5	94.1	100.3
7	97.9	95.5	100.1
8	99.7	94.7	100.0
9	97.0	95.8	100.3
10	97.5	93.2	101.7
11	94.0	97.0	101.0
12	101.2	95.1	103.9

$K = 3$ Factors			
N	G -CEXCH	$G(I_\lambda)$ -CEXCH	G -PSO
10	93.1	95.4	101.6
11	92.9	96.9	104.2
12	90.7	90.3	103.8
13	92.9	99.9	103.2
14	87.6	100.0	100.5
15	98.5	100.1	102.5
16	100.1	100.2	108.1

Table 4.2: Number of function evaluations across the $n_{\text{run}} = 140$ G -PSO runs using $S = 150$ particles.

$K = 1$ Factor			
N	niter	# f evaluations	$\log_{10}(\text{# } f\text{evaluations})$
3	6671	1000650	6.00
4	22843	3426450	6.53
5	31949	4792350	6.68
6	11210	1681500	6.23
7	32282	4842300	6.69
8	38485	5772750	6.76
9	16927	2539050	6.40
$K = 2$ Factors			
N	niter	# f evaluations	$\log_{10}(\text{# } f\text{evaluations})$
6	81714	12257100	7.09
7	81341	12201150	7.09
8	73388	11008200	7.04
9	87649	13147350	7.12
10	97013	14551950	7.16
11	110911	16636650	7.22
12	104700	15705000	7.20
$K = 3$ Factors			
N	niter	# f evaluations	$\log_{10}(\text{# } f\text{evaluations})$
10	182429	27364350	7.44
11	216165	32424750	7.51
12	233427	35014050	7.54
13	229837	34475550	7.54
14	232699	34904850	7.54
15	218081	32712150	7.51
16	239812	35971800	7.56

Table 4.3: New two-factor G -optimal designs generated by PSO. G -scores are shown for PSO designs and for corresponding GA generated designs.

$K = 2$		$N = 6$	
$G_{\text{PSO}} = 75.0304$		$G_{\text{GA}} = 74.7848$	
$G_{\text{eff}} = 100.3$			
x_1	x_2		
0.170301	-0.21245		
-0.50794	0.999999		
-0.85572	-0.96913		
0.997798	0.830996		
1	-1		
-1	0.536392		

$K = 2$		$N = 7$	
$G_{\text{PSO}} = 80.2387$		$G_{\text{GA}} = 80.1917$	
$G_{\text{eff}} = 100.1$			
x_1	x_2		
1	-0.61253		
0.999956	0.924872		
0.025791	0.772488		
-1	1		
-0.77382	-0.03485		
-0.95239	-1		
0.594303	-1		

$K = 2$		$N = 9$	
$G_{\text{PSO}} = 86.6336$		$G_{\text{GA}} = 86.3495$	
$G_{\text{eff}} = 100.3$			
x_1	x_2		
0.435453	-1		
-1	-1		
-1	1		
1	-1		
1	1		
-1	-0.43544		
1	0.435453		
7.70E-05	0.000021		
-0.43544	1		

$K = 2$		$N = 10$	
$G_{\text{PSO}} = 87.4032$		$G_{\text{GA}} = 85.9373$	
$G_{\text{eff}} = 101.7$			
x_1	x_2		
-1	0.260147		
0.999957	-0.6179		
1	0.540804		
-0.53865	-1		
-1	1		
0.999406	-1		
0.096459	-0.16578		
1	1		
-1	-1		
-0.28952	1		

$K = 2$		$N = 11$	
$G_{\text{PSO}} = 87.0703$		$G_{\text{GA}} = 86.2093$	
$G_{\text{eff}} = 101$			
x_1	x_2		
-0.6564	-1		
0.444013	0.998583		
-1	-1		
1	0.36528		
1	0.999999		
0.688845	-0.99926		
1	-1		
-0.92361	-0.03252		
-0.69015	0.976289		
-1	0.999998		
0.022237	-0.04057		

$K = 2$		$N = 12$	
$G_{\text{PSO}} = 88.1719$		$G_{\text{GA}} = 84.8966$	
$G_{\text{eff}} = 103.9$			
x_1	x_2		
-0.13934	0.00247		
-1	0.999807		
1	-1		
-1	-1		
0.258812	0.99998		
1	0.629227		
-0.99999	0.999934		
0.999989	-0.64716		
-0.89939	0.007285		
0.213116	-0.99956		
-1	-0.96893		
0.999999	0.999999		

Table 4.4: New three-factor G -optimal designs generated by PSO for $N = 10, 11, 12, 13$. G -scores are shown for PSO designs and for corresponding GA generated designs.

$K = 3$			$N = 10$		
$G_{\text{PSO}} = 71.4253$			$G_{\text{GA}} = 70.267$		
			$G_{\text{eff}} = 101.6$		
x_1	x_2	x_3			
0.123168	-1	-1			
0.999999	-1	-0.23097			
0.999999	0.647752	-1			
-1	0.839207	0.99998			
0.020484	0.273614	0.13189			
-1	-0.51107	-0.90444			
0.797262	-0.74605	1			
0.999832	1	0.82643			
-0.76012	1	-0.88557			
-0.88334	-1	0.744738			

$K = 3$			$N = 11$		
$G_{\text{PSO}} = 80.5095$			$G_{\text{GA}} = 77.2634$		
			$G_{\text{eff}} = 104.2$		
x_1	x_2	x_3			
-0.99999	0.999875	1			
0.972844	-0.97778	-0.84772			
-0.95013	0.851057	-1			
0.016123	1	-0.32245			
-0.87117	-0.99999	0.95413			
0.999999	0.724464	-1			
-0.72417	-1	-1			
-1	-0.3122	-0.08383			
0.284597	0.008598	1			
0.999998	0.999996	0.851942			
0.999979	-0.98313	0.806747			

$K = 3$			$N = 12$		
$G_{\text{PSO}} = 83.349$			$G_{\text{GA}} = 80.2657$		
			$G_{\text{eff}} = 103.8$		
x_1	x_2	x_3			
0.057224	-1	0.034209			
-1	0.01607	0.052685			
1	-0.97614	0.874472			
-1	-0.99993	0.999999			
1	0.91264	-0.79712			
0.636813	0.164705	1			
0.999958	-0.84258	-1			
0.919046	1	0.739721			
-1	-1	-1			
-0.83952	1	0.999999			
-1	0.999194	-0.83791			
-0.007	0.620902	-1			

$K = 3$			$N = 13$		
$G_{\text{PSO}} = 86.4558$			$G_{\text{GA}} = 83.7388$		
			$G_{\text{eff}} = 103.2$		
x_1	x_2	x_3			
0.121603	0.231202	0.999996			
-1	-0.99497	-1			
-1	-0.14928	0.124691			
0.208365	-1	-0.22702			
0.535044	1	-1			
-1	-1	1			
0.999475	-1	0.999997			
-1	0.999999	0.994687			
0.999939	0.999994	-0.04835			
0.870509	-1	-0.85421			
0.998993	-0.01356	-0.99999			
-0.98701	0.951149	-0.94818			
0.88939	0.858407	0.999991			

Table 4.5: New three-factor G -optimal designs generated by PSO for $N = 14, 15, 16$. G -scores are shown for PSO designs and for corresponding GA generated designs.

$K = 3$			$N = 14$		
$G_{\text{PSO}} = 89.7063$			$G_{\text{GA}} = 89.2857$		
			$G_{\text{eff}} = 100.5$		
x_1	x_2	x_3			
0.09943	0.06159	-1			
-1	1	1			
1	-0.23	0.109736			
-1	0.302402	-0.0649			
1	-1	-1			
-1	-0.99939	-1			
0.285617	1	0.098852			
-0.22231	-1	-0.17194			
-1	-1	1			
0.999999	1	1			
-0.13479	-0.15759	1			
1	-1	1			
-1	0.999989	-1			
1	1	-1			

$K = 3$			$N = 11$		
$G_{\text{PSO}} = 85.993$			$G_{\text{GA}} = 83.9161$		
			$G_{\text{eff}} = 102.5$		
x_1	x_2	x_3			
-0.27061	-0.00304	-1			
-1	1	-1			
-0.1802	1	0.121891			
-1	-1	-1			
-0.2255	-0.07407	1			
-1	-1	1			
0.999999	0.046682	-0.64352			
0.984381	1	-1			
0.999999	0.014207	0.519179			
-1	1	1			
0.999999	1	1			
-1	0.0224	-0.05955			
1	-1	-1			
1	-1	1			
-0.16515	-1	0.064085			

$K = 3$			$N = 16$		
$G_{\text{PSO}} = 85.7876$			$G_{\text{GA}} = 79.3651$		
			$G_{\text{eff}} = 108.1$		
x_1	x_2	x_3			
-1	1	-1			
-0.14365	-0.50543	-1			
-1	-1	-1			
0.999996	0.999999	-1			
1	-1	0.353865			
0.444126	0.999763	0.153807			
0.994695	0.032632	-0.93571			
0.293814	-1	1			
1	0.999954	1			
-1	-1	0.97247			
-1	0.999988	1			
-0.65291	0.34445	0.999893			
1	-0.38829	1			
-0.56818	0.907351	-0.73162			
0.908731	-1	-1			
-1	-0.45915	0.024148			

Table 4.6: G -efficiencies of published G -optimal designs for $K = 4, 5$ factors. The last column is the relative efficiency of the best G -PSO design compared to previously published designs. In all cases, PSO searches generated as or more efficient G -optimal designs (with some cases being much more efficient).

$K = 4$ Factors				
N	source	CEXCH	G -PSO	G -releff
15	Rodriguez (2010)	48.89	71.09	145.41
17	Hernandez (2018)	70.14	73.90	105.36
20	Rodriguez (2010)	65.11	80.20	123.18
24	Rodriguez (2010)	81.05	85.95	106.05
$K = 5$ Factors				
N	source	CEXCH	G -PSO	G -releff
21	Rodriguez (2010)	38.74	68.67	177.26
23	Hernandez (2018)	73.02	73.19	100.24
26	Rodriguez (2010)	72.47	75.31	103.92
30	Rodriguez (2010)	75.80	76.16	100.47

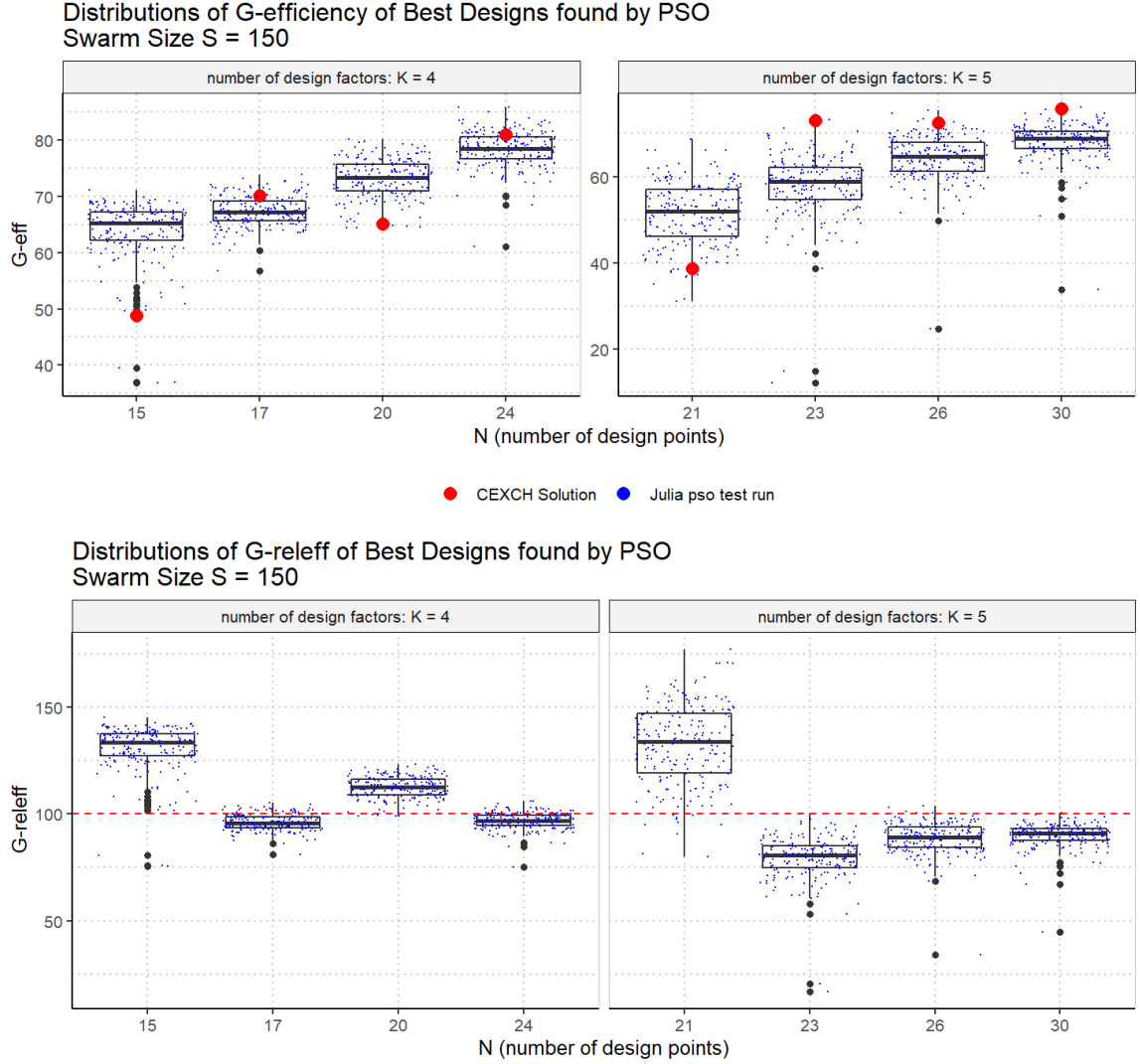


Figure 4.3: The top set of panels shows boxplots of the $n_{\text{run}} = 210$ G -efficiencies of the designs generated by PSO. The red dot represents the G -eff of the coordinate exchange algorithms generated design from Rodrigues et. al. (2010) and Hernandez and Nachtsheim (2018). The bottom set of panels show boxplots of the G -releff of the PSO generated designs with the G -score of the coordinate exchange generated designs used in the denominator of the relative efficiency calculation.

Table 4.7: The G -PSO G -optimal design for $K = 4$, $N = 15$.

$$G_{\text{PSO}} = 71.09 \quad G_{\text{CEXCH}} = 48.89 \quad G_{\text{releff}} = 145.41$$

x_1	x_2	x_3	x_4
0.302935	0.887481	0.286558	0.423355
-0.98981	-1	0.969024	0.494841
1	1	-0.98361	1
-1	0.999999	0.441658	-1
1	-0.53206	-1	-0.27309
0.98395	-1	0.472542	-1
0.309884	-1	-1	1
-0.41944	-0.36411	1	-0.99999
1	1	1	-0.7915
-1	-1	-1	-1
-1	-0.32967	-0.45073	1
-0.97123	1	-1	0.410811
1	-0.60809	1	1
-0.88136	1	1	1
0.55599	0.899941	-1	-1

Table 4.8: The G -PSO G -optimal design for $K = 4$, $N = 17$.

$$G_{\text{PSO}} = 73.90 \quad G_{\text{CEXCH}} = 70.14 \quad G_{\text{releff}} = 105.36$$

x_1	x_2	x_3	x_4
-0.6593	-1	1	1
0.999998	-0.27492	1	-0.99983
-0.65525	0.999994	-1	1
-1	0.142802	0.10533	0.999951
1	0.999966	-0.07395	-1
-0.14056	0.895821	0.999999	-0.99557
-0.53151	-0.30855	-1	-0.84341
0.999998	1	0.983932	0.999998
0.999805	-1	-1	-0.99999
1	-0.77773	-0.86869	1
0.999998	-1	0.892947	0.307346
-1	0.999999	-1	-0.99999
0.999641	0.903413	-1	0.169206
-1	-1	-1	0.646097
-1	-1	0.743172	-1
0.26792	-1	0.053794	-0.23032
-1	0.965609	0.99999	0.137019

Table 4.9: The G -PSO G -optimal design for $K = 4$, $N = 20$.

$$G_{\text{PSO}} = 80.20 \quad G_{\text{CEXCH}} = 65.11 \quad G_{\text{releff}} = 123.18$$

x_1	x_2	x_3	x_4
1	1	-1	0.317435
-1	-1	0.999996	-1
0.999975	-1	-1	-1
-1	0.999999	-0.09412	-1
-0.45122	-1	-1	0.999999
-1	-1	0.999993	0.999997
-1	1	1	-0.41959
0.998183	0.991994	0.999999	-1
-1	0.176938	0.999833	1
-1	0.999999	-1	0.999977
-1	-0.56141	-1	-0.91253
-0.99984	-0.8648	-0.4356	0.068839
0.147577	0.999991	0.10737	1
-0.0914	0.999999	-1	-1
0.999986	-1	0.937599	0.808684
0.999946	1	0.999861	0.999999
0.999995	0.490758	-0.17919	-1
0.785769	-0.98836	0.703881	-0.84591
0.166995	-0.07194	0.999932	-0.28523
1	-0.54443	-1	1

Table 4.10: The G -PSO G -optimal design for $K = 4$, $N = 24$.

$$G_{\text{PSO}} = 85.95 \quad G_{\text{CEXCH}} = 81.05 \quad G_{\text{releff}} = 106.05$$

x_1	x_2	x_3	x_4
0.999989	-1	0.999998	-0.953025
0.999993	1	-0.999871	0.999999
-1	0.082288	1	0.007329
-0.184423	-0.086385	-0.916944	-0.999993
-0.699209	0.999993	-1	0.458423
1	0.999993	-0.999987	-0.975958
-0.53179	-0.869675	1	-0.999999
0.998729	-1	-0.999996	0.946564
-0.999999	0.999982	-0.999999	-0.999991
0.350613	0.946408	-0.043458	-0.204128
-1	-0.999999	-0.113837	-1
-0.999998	1	0.180651	0.999999
-1	0.993105	1	-0.999473
-1	0.014335	-1	1
-0.25014	0.999999	1	0.999995
-1	-0.999999	0.999067	0.976727
0.999993	-1	-0.999996	-0.962068
0.998262	1	0.959943	-0.999999
0.842343	-0.364967	0.158512	-0.036263
0.787106	-0.999995	0.970295	0.840789
0.999955	-0.208088	0.372257	0.999999
1	1	1	0.999998
-0.515707	-1	-0.750875	1
-1	-0.999998	-1	-0.361417

Table 4.11: The G -PSO G -optimal design for $K = 5$, $N = 21$.
$$G_{\text{PSO}} = 68.67 \quad G_{\text{CEXCH}} = 38.74 \quad G_{\text{releff}} = 177.26$$

x_1	x_2	x_3	x_4	x_5
-0.64816	0.868462	0.999999	1	-0.94019
0.315014	-0.27526	-0.68548	-1	-1
-1	-0.7853	0.927942	-0.78191	-1
-0.75568	-1	-0.99995	0.993406	-0.99656
-0.99997	-0.14602	0.04158	0.857948	-0.02095
1	-0.50864	0.999896	-0.99874	0.999997
-0.96863	0.993964	0.882819	-0.98812	0.974981
1	0.97446	0.953867	0.895293	0.93321
-0.97336	1	-0.78653	0.999999	0.999626
0.988127	0.999991	0.999743	-0.88992	-0.94298
-0.2295	0.242609	-1	0.030758	0.685815
0.99077	1	-0.68517	-0.99916	0.999679
-0.99407	0.992837	-1	-1	-0.43773
-0.99998	-0.99931	-0.94296	-1	0.999999
-0.87786	1	-0.24226	-0.00159	-1
-0.85335	-1	0.999997	0.791091	0.979707
0.012147	-1	0.619007	-0.86432	0.080026
1	-1	-1	-0.66666	-0.59016
0.999272	-0.97255	0.761585	0.848943	-0.99986
0.926485	-1	-0.80197	0.999999	0.999889
0.999998	0.8844	-0.97365	0.996343	-0.80788

Table 4.12: The G -PSO G -optimal design for $K = 5$, $N = 23$.
$$G_{\text{PSO}} = 73.19 \quad G_{\text{CEXCH}} = 73.02 \quad G_{\text{releff}} = 100.24$$

x_1	x_2	x_3	x_4	x_5
0.999992	-1	0.928967	0.999997	-0.67445
-1	0.999999	-0.98461	0.778856	1
0.999823	0.365964	0.999998	0.23501	-0.80801
0.999998	-0.99999	-0.99994	0.508559	1
-0.98424	-1	-1	-0.99987	0.997717
-0.99999	0.999998	0.999997	-1	0.999964
0.802311	-0.44046	1	-0.76101	0.999996
0.899579	1	0.788864	-0.99999	-1
0.999984	0.999997	-0.9714	-0.36282	0.18479
0.96494	-1	0.433358	-0.99999	0.4766
-0.0582	1	0.999998	0.793829	0.197345
-0.99997	0.683957	0.86565	0.995626	-1
0.039575	-1	0.028443	0.099622	-0.99995
1	-0.67349	-1	-0.99997	-1
0.333113	-0.36516	-0.99926	0.999878	0.769817
-1	0.216219	-0.06096	-0.92437	0.341512
1	0.995466	0.533615	0.999991	0.999979
0.495033	0.999984	-1	-0.99997	0.999986
-1	1	-0.99994	-0.93073	-0.99996
-0.96094	-1	0.900887	0.91178	0.999995
-0.92244	-0.99998	1	-1	-0.88111
-0.99999	-0.99999	-0.99999	0.999997	-0.928
0.767233	0.998257	-0.88805	1	-1

Table 4.13: The G -PSO G -optimal design for $K = 5$, $N = 26$.
$$G_{\text{PSO}} = 75.31 \quad G_{\text{CEXCH}} = 72.47 \quad G_{\text{releff}} = 103.92$$

x_1	x_2	x_3	x_4	x_5
-1	1	0.935337	-0.99972	-0.99993
0.912917	0.999998	0.999995	0.999698	-0.99992
-0.99999	-0.65416	-0.49504	-0.5608	-0.99893
0.90901	-0.99976	-1	-0.69599	0.735379
0.99999	0.050243	0.935778	-0.47541	-0.02692
1	0.999988	-0.84703	-1	-0.99977
-0.88466	-1	-1	0.999999	0.590294
0.785925	-1	1	-1	-0.99991
-0.99905	-0.92192	0.999866	0.999971	-0.99665
-0.84071	-0.67866	-0.99998	0.160719	1
0.004889	0.999991	-0.46745	0.081422	-0.22566
0.0557	-0.21894	-1	-0.99606	0.458947
0.983515	-0.99993	0.999952	1	0.999994
0.999232	-0.28586	-0.99993	0.994551	-0.99961
0.071395	-0.9978	-0.99999	0.079878	-1
0.775487	0.99998	0.999507	-0.99819	1
-0.99842	-0.62494	0.711922	-0.36159	1
-1	0.999999	-0.93652	-0.94575	0.999848
0.999979	-0.99965	-0.10418	-0.99828	0.998719
0.018039	0.057149	0.014752	0.999984	0.999804
1	-0.99993	-0.19361	0.991039	-0.57898
0.999986	0.999987	-0.96564	0.901591	0.997256
-1	1	0.999996	0.999907	0.911052
-1	-0.99881	-0.9021	-0.99983	-0.77096
-0.99954	0.999929	-0.99987	1	-1
-0.70849	-0.99949	1	-1	0.652671

Table 4.14: The G -PSO G -optimal design for $K = 5$, $N = 30$.
$$G_{\text{PSO}} = 76.16 \quad G_{\text{CEXCH}} = 75.80 \quad G_{\text{releff}} = 100.47$$

x_1	x_2	x_3	x_4	x_5
1	0.304368	0.999998	1	0.23828
-0.99984	0.038145	-1	1	-0.99998
0.999985	1	0.999868	0.045942	0.999996
1	0.999987	-1	-0.72697	1
0.951284	0.055656	-0.99998	-1	0.913188
-1	-1	0.996986	0.999998	1
-1	-1	-1	-1	-0.74812
1	-0.99809	0.016051	1	0.157196
0.999986	-0.99994	-1	-0.73955	-1
0.255602	0.904926	0.189544	1	1
1	1	-1	1	-0.99997
0.986066	0.243414	0.157757	-0.99966	-1
0.015095	-0.99999	0.999999	-0.89748	0.371031
-1	-0.01595	-0.91993	-1	1
-1	0.999984	0.893253	1	-0.01112
-0.92083	-0.99998	1	-1	-0.99975
0.999975	1	1	-0.99909	-0.75769
1	-1	-1	0.999986	0.999965
-0.10026	-0.82407	-0.99995	1	-0.09713
0.999998	-1	0.999999	0.865323	-1
0.209342	-1	0.999999	0.561547	0.999823
-0.99643	1	-1	-0.99999	-1
-0.99995	0.999939	-0.99992	0.999663	1
-0.99992	-1	-0.72458	-0.34826	1
0.226162	0.95362	-0.68084	-0.11698	-0.17403
1	-0.99998	0.568002	-1	1
-0.92815	-1	-0.08238	1	-0.9996
-0.10201	1	0.999966	0.999915	-1
-0.99999	0.431681	0.871153	-0.14252	-0.90278
-0.9409	1	1	-1	0.999998

CHAPTER FIVE

A PARTICLE SWARM BASED ALGORITHM FOR GENERATING
IV-OPTIMAL DESIGNS WITH SPECIFIED REPLICATION STRUCTUREABSTRACT

In this chapter we explore the efficacy of PSO for generating *IV*-optimal designs. *IV*-optimal designs are those designs that minimize the average prediction variance over $\mathcal{X}$. As such, these designs are popular in practice when the objective of the experiment is to predict the response in unsampled areas of $\mathcal{X}$. We implement a version of the *IV*-criterion which does not depend on N and so the relative efficiency of designs of different sample sizes can be directly compared. We implement PSO on all $\{K, N\}$ scenarios studied by Borkowski (2003). We extend these results to small exact response surface designs for $K = 4$ factors. Then, following a call from Jensen (2018), we develop an approach that allows the user to specify a replication structure. We employ this approach on a number of $\{K, N\}$ scenarios and discuss the results and their implication on the data analysis that ensues experimentation. This discussion focuses on the benefit of replication in design of experiments, the decomposition of residual error into pure error and lack-of-fit residual components, and a full derivation of the lack-of-fit test statistic and power calculation.

5.1 Introduction

IV-criterion stands for *integrated variance* criterion. As such *IV*-optimal designs focus on minimizing the average prediction variance over $\mathcal{X}$. Due to this and to the relative ease of computational complexity (as compared to other prediction variance criterion, specifically the *G*-criterion) authors such as [6, 44, 51, 68] indicate

that these designs have achieved widespread popularity among practitioners of DoE, especially for the response surface modeling phase of experiments wherein minimizing uncertainty of the prediction of the mean response over $\mathcal{X}$ is of paramount importance. Therefore, a comprehensive study of the ability and efficacy of PSO to generate *IV*-optimal designs represents an important contribution to the statistics literature.

Our first goal in this chapter is to demonstrate that PSO accurately and efficiently (with respect to computing time and with a small number of runs of the algorithm) generates *IV*-optimal designs. To accomplish this goal we first apply PSO to all design scenarios reported in Borkowski (2003) [13]. Then, we extend the results of that paper to $K = 4$ factors, covering all cases described in Chapter 4 of this dissertation. We then compare the results of the *IV*-efficient designs to the *G*-efficient designs on both criterion and via *fraction of design space* plots.

A recent publication by Jensen (2018) [51] in *Quality Engineering* titled “Open problems and issues in optimal design” discusses the popularity of the optimal design paradigm in contemporary statistical practice and highlights some current areas for improvement. Specifically, in the context of discussing application of *IV*-optimal designs, Jensen highlights that the focus only on the optimality criterion score of the design has produced an under valuation in the literature regarding the inclusion of pure experimental replicates in the design. Jensen highlights the following benefits of including pure experimental replicates:

1. An estimate of the residual error variance σ^2 is required for the analysis of the model. The estimate of this term propagates into all inferences made, including: overall model significance, estimates of the standard errors of the coefficient estimates, lack-of-fit (LOF) tests, and uncertainty estimates of predictions made from the fitted model. As such, inference is generally improved if the error variance estimate is based on pure-error (i.e., derived from the experimental

replicates) as opposed to a model-based estimate of error (which occurs when no pure experimental replicates are included in the design). Jensen cites Gilmour and Trinca (2012) and indicates that a bias generally occurs in estimating σ^2 if the estimate is based on model-based error residuals [43].

2. Jensen highlights that many optimal design algorithms do not generate replicates of existing points as the sample size increases (with the exception of replicates at the center of the design region). Instead, addition of new design points are favored by the optimality criterion and this condition presents challenges in practice as experiments with a large number of unique points can be costly (e.g., the settings on the experimental system have to change with each run, and ensuring this is done correctly can be costly in time or effort). Thus, the ability for the designer to be able to specify some number of replicated design points is an important aspect of optimal design generation that could benefit from more research.
3. Detection of outliers, and a robustness of the modeling to those outliers, is another important reason to include pure experimental replicates. Jensen cites a number of papers (e.g., Sanders and Hild (2012) [80]) which indicate that outlier detection using traditional model diagnostics with an experiment absent of pure experimental replicates is a difficult task.
4. Provided the number of design points N is large enough, the Central-Composite Design (CCD) is a common classical design that includes center replicates. On the hypercube $\mathcal{X}$, the CCD includes all vertices, axial points which correspond to edge or face center points, and N_C replicates of the center point $\{0\}^K$. Jensen calls for more research into efficient uses of pure replication [18].

Thus our second goal in this chapter is to produce a methodology for producing PSO-generated *IV*-efficient designs which allows the user to specify the maximal number of unique points (and hence a maximal number of pure replicates) in the design generation problem via an adjustment to the optimality criterion. This requires some new notation. Let M represent the (maximum) number of unique design points the user may specify, N_i represent the number of replicates of design point i , and we have that the total number of design points is $N = \sum_{i=1}^M N_i$. We apply our methodology to several design scenarios for $K = 2, 3, 4$ factors. Further, as one of the benefits of including pure replicates is the ability to conduct the LOF test, we provide a full derivation of the null and alternative distribution of the LOF statistic, and provide a power analysis for select replication structures. We include the resulting information in the discussion of the design formulation problem which includes specification of the number of replicates to include in the design.

5.2 The *IV*-criterion

As we illustrated in Chapter 2, design criteria are often constructed using the average model information matrix

$$\mathbf{M}(\mathbf{X}) = \frac{\mathbf{F}'\mathbf{F}}{N}.$$

Eq. 2.17 shows that this formulation of the *IV*-criterion is a factor of N and thus average information *per design point*. Recent authors such as [44, 51, 68] highlight that using the average model information matrix to compute the *IV*-criterion does not allow for comparison of the candidate *IV*-optimal design across sample sizes N and so a non-scaled version of the criteria which uses the total model information

matrix

$$\mathbf{M}(\mathbf{X}) = \mathbf{F}'\mathbf{F}$$

is often preferred in practice. Thus, the form of the IV -criterion that we apply in this chapter is

$$IV(\mathbf{X}) = \frac{1}{V} \text{tr} \left\{ (\mathbf{F}'\mathbf{F})^{-1} \mathbf{W} \right\} \quad (5.1)$$

where V is the volume of $\mathcal{X}$, and

$$\mathbf{W} = \int_{\mathcal{X}} \mathbf{f}(\mathbf{x}') \mathbf{f}'(\mathbf{x}') d\mathbf{x}' \quad (5.2)$$

is often referred to as the *region moments* matrix [44, 68]. Provided that $\mathcal{X}$ is an ‘easy’ geometry to work with, it is possible to compute V and $\mathbf{W}$ exactly. When $\mathcal{X} = [-1, 1]^K$ (i.e., the K -dimensional hypercube) we have $V = 2^K$ and Goos and Jones (2011) [44] provide the following general formula for W of arbitrary dimension K for the second order model:

$$\mathbf{W} = 2^K \begin{pmatrix} 1 & \mathbf{0}'_K & \mathbf{0}'_{\frac{K(K-1)}{2}} & \frac{1}{3} \mathbf{1}'_K \\ \mathbf{0}_K & \frac{1}{3} \mathbf{I}_K & \mathbf{0}_{K \times \frac{K(K-1)}{2}} & \mathbf{0}_{K \times K} \\ \mathbf{0}_{\frac{K(K-1)}{2}} & \mathbf{0}_{\frac{K(K-1)}{2} \times K} & \frac{1}{9} \mathbf{I}_{\frac{K(K-1)}{2}} & \mathbf{0}_{\frac{K(K-1)}{2} \times K} \\ \mathbf{1}_K & \mathbf{0}_{K \times K} & \mathbf{0}_{K \times \frac{K(K-1)}{2}} & \frac{1}{5} \mathbf{I}_K + \frac{1}{9} (\mathbf{J}_K - \mathbf{I}_K) \end{pmatrix} \quad (5.3)$$

where $\mathbf{0}_K$ is an $K \times 1$ vector of 0’s, $\mathbf{0}_{K \times K}$ is an $K \times K$ matrix of 0’s, $\mathbf{1}_K$ is an $K \times 1$ vector of 1’s, $\mathbf{J}_K$ is an $K \times K$ matrix of 1’s, and $\mathbf{I}_K$ is an $K \times K$ identity matrix.

If the design space $\mathcal{X}$ is a geometry where the required integrals are difficult to compute, the V , $\mathbf{W}$ and hence the IV -criterion may all be approximated via Monte-Carlo methods.

Note that on the IV -scale as shown in Eq. 5.1, smaller IV -score indicates a better design.

5.3 PSO-generated IV -optimal designs

5.3.1 Reproducing Borkowski (2003) Genetic Algorithm Generated Designs

As a first step to using PSO to generate IV -optimal designs, we cover the seven N scenarios for each $K = 1, 2, 3$ number of design factors as studied in Borkowski (2003) and as shown in Chapter 3 Table 3.2. In this study we implemented $n_{\text{run}} = 140$ PSO searches per design scenario using $S = 150$ particles per search. The runs per design scenario were parallelized over seven computing cores. The total run-time to complete all searches presented in this section was less than 4 minutes. This run-time is an encouraging result as it speaks favorable toward the efficiency and applicability of using PSO to generate candidate IV -optimal designs for a practitioner in near real time. We present distribution plots of the results in Figure 5.1. In all cases it can be seen that the PSO-generated design met or improved the IV -score of the GA generated design. In three cases, $K = 1, N = 6$, $K = 3, N = 13$, and $K = 3, N = 14$ PSO found a design with a slightly better IV -score (relative efficiencies are over 100% by 0.5-1% for these design scenarios).

The results presented in this section validate the ability of PSO to produce highly IV -optimal designs in a computationally fast and efficient way (e.g., requiring only a few hundred searches).

5.3.2 $K = 4$ factor IV -optimal Designs

We implemented PSO searches for $K = 4$ factor IV -optimal designs for the same scenarios studied on the G -criterion in Chapter 4: $N = 15, 17, 20$, and 24. We ran $n_{\text{run}} = 210$ searches with $S = 250$ particles parallelized over 7 computing cores. The entire set of searches took approximately 26 minutes to complete. The formula for the IV -criterion we applied in this set of searches was that in Eq. 5.1 which utilizes

the total design information and hence we can compare IV -scores across different N . We present boxplots of the result of the searches in Figure 5.2. The top panel shows the distributions of IV -scores of the PSO searches. As we applied the version of the IV -criterion which is not scaled by N , the reader may compare across N (smaller IV -criterion indicates a better design). In the middle panel we show the IV -efficiency using the global minimum IV -efficiency found across all N cases. Thus, this panel illustrates the added benefit to the researcher, in terms of IV -efficiency, for including 2 to 3 extra points in the design (each N case shows approximately a 12-15% increase in efficiency for adding 2 to 3 extra points to the design). Last, in the bottom panel, we show the IV -efficiencies computed for each N scenario conditioned on that scenario. The graphic illustrates that PSO starts to have a harder time finding the global minimum for the $N = 20, 24$ cases.

We report the IV -PSO generated designs and their IV -scores in Tables 5.1, 5.2, 5.3, and 5.4 for $N = 15, 17, 20$ and 24 respectively.

5.3.3 Comparison of the properties of the $K = 4$ PSO IV -optimal designs to the PSO G -optimal designs.

The IV and G -criteria are based on the scaled prediction variance, and therefore are useful to the researcher if the goal after data collection is to predict the value of the response, and its uncertainty, at new (unsampled) points in $\mathcal{X}$. Therefore a cross-comparison of the properties of the IV and G -optimal designs is useful to understand in what sense each design may be ‘better’.

We will approach this comparison via three quantitative analyses:

1. A comparison of both designs on both the IV and G -scores.
2. A comparison of both designs via the *fraction of design space* (FDS) plot.

3. A comparison of both designs via the *conditional fraction of design space* (CFDS) plot.

First, we show the comparison of the IV and G scores for the designs in Table 5.5. For each N considered, the table illustrates that an IV -optimal design sacrifices a lot on the G -efficiency, and vice-versa. The tradeoff is not always equal, for example, for the $N = 15$ case the G -PSO design only loses about 29% efficiency on the IV -score relative to the IV -PSO design, whereas the IV -PSO design loses about 48% efficiency on the G -score relative to the G -PSO design. The trade-off in efficiency is closer for the $N = 20$ scenario.

While the scoring on the criteria represent a first comparison of the properties of two-candidate designs, only a single-valued criteria is considered, so this obscures features of each design on the *scaled-prediction variance* (SPV) surface over $\mathcal{X}$ [17]. Thus, a second comparison of candidate designs SPV structure on $\mathcal{X}$ often used in practice is the *fraction of design space* (FDS) plot [15]. The FDS plot is a representation of the SPV using the formula

$$SPV(\mathbf{x}' \in \mathcal{X}, \mathbf{X}) = \mathbf{f}'(\mathbf{x}') (\mathbf{F}'\mathbf{F})^{-1} \mathbf{f}(\mathbf{x}') \quad (5.4)$$

and describes the fraction of $\mathcal{X}$ that has SPV less than or equal to some specific value v^* [44]. We present the FDS plot comparison of the G -PSO and IV -PSO designs for each N scenario in Figure 5.3. The SPV is not scaled by N , and so one can assess the relative impact of increasing N by comparing the general drop in SPV across the panels. The figure illustrates that the new G -PSO designs have a larger prediction variance than the IV -PSO designs for a very large fraction of the design space $\mathcal{X}$ (being approximately 96% of $\mathcal{X}$ for $N = 15$ and nearly 99% of $\mathcal{X}$ for the other N scenarios). This suggests that the IV -PSO designs would be preferred if prediction

is the end goal of the statistical modeling.

While the FDS plot is a useful tool to compare SPV properties of candidate designs, it does not indicate where in $\mathcal{X}$ a design has better SPV. Thus, we implemented conditional-SPV (CSPV) plots on each scenario. The CFDS plot shows SPV distribution on each dimension x_i along with an envelope of minimum and maximum SPV and the mean SPV. We present CSPV graphics in Figures 5.4, 5.5, 5.6, and 5.9 for the $N = 15, 17, 20$ and 24 scenarios, respectively. The general story told by these graphics indicates that the G -criterion has a tendency to produce designs with a uniform SPV profile over $\mathcal{X}$ at the sacrifice of the mean SPV, and while the IV -criterion produces designs that minimize the mean SPV, it can be seen that the IV -criterion has a tendency to greatly sacrifice the maximum SPV, and this can be seen to occur at the vertices of $\mathcal{X}$.

The observations made here regarding the properties of the $K = 4$ factor G -PSO and IV -PSO optimal designs, specifically that the IV -optimal designs have lower SPV over the majority of $\mathcal{X}$, are consistent with what has been demonstrated by previous authors for $K = 1, 2, 3$ factors [44, 78]. For this reason, the literature has tended to focus and favor IV -optimal designs in practice when the post-data analysis goal is prediction [44, 51]. Thus, a general similar recommendation can be made in favor of the IV -PSO designs vs. the G -PSO designs for prediction as well. However, we contend that, while the SPV has been used as a tool to make these recommendations when analyzing competing designs, the SPV graphic, only helps with the “how much better” question and not the “where is it better” question. This is where the CSPV plots prove useful.

To illustrate the usefulness of the CSPV plot as a next step in analyzing competing designs after looking at the FDS plot, we will continue with the $K = 4$, $N = 24$ design scenario. First we note that the FDS plot (lower right) of Figure 5.3

illustrates that the *IV*-PSO design has smaller prediction variance than the *G*-PSO design over 99% of $\mathcal{X}$. Next, we consider the CSPV plot of these two designs in Figure 5.9. Each panel is a projection of the SPV surface from 4-dimensions into 1-dimension. The *G*-PSO design has an apparent symmetric quadratic SPV over $\mathcal{X}$ (each projection onto x_i looks very similar). The corresponding *IV*-PSO CSPV plots, in contrast, show some asymmetry. Specifically, when $x_3 = -1$ and $x_4 = 1$ the *IV*-PSO design has larger SPV than it does for other values of these variables. Using this information, we can focus in on that specific face of the 4-dimensional hypercube (which has 24 faces). Thus, we fix $x_3 = -1$ and $x_4 = 1$ and produce contour plots of the SPV surface as a function of x_1 and x_2 which are presented in Figure 5.8. These graphics indicate that the *G*-PSO design tends to have a lower SPV than the *IV*-PSO design over the majority of this face. To quantify this, we present a conditional-FDS (CFDS) plot for this face in Figure 5.9. In this graphic it can clearly be seen that the *G*-PSO design has a uniformly smaller SPV than the *IV*-PSO design on the $(x_3 = -1, x_4 = 1)$ face of $\mathcal{X}$. This illustration shows that if prediction on the boundaries of $\mathcal{X}$, and specifically at the extreme face $(x_3 = -1, x_4 = 1)$, are very important, than the *G*-optimal design should be preferred.

5.4 The importance of pure replication in designed experiments

Several benefits to including pure experimental replicates in the design were briefly discussed in the introduction. All of these focus, in some way, on the OLS assumption that model errors are subject to a constant variance σ^2 over $\mathcal{X}$. Thus, an obvious benefit in including pure replicates at different points of $\mathcal{X}$ is to assess the homoscedasticity assumption. A stronger benefit, however, is that experimental replicates enable a ‘clean’ estimate of the error variance σ^2 .

The estimate $\hat{\sigma}^2$ is central and of primary importance to linear model analysis.

This estimate propagates into all uncertainty calculations made during inference. It is a component of the standard errors of the estimates of the linear model coefficients

$$\widehat{\text{Var}}(\hat{\boldsymbol{\beta}}) = \hat{\sigma}^2 (\mathbf{F}'\mathbf{F})^{-1} \quad (5.5)$$

and thus is used to detect and quantify factor effects. The estimate is also a primary component in the estimated prediction variance

$$\widehat{\text{Var}}(\hat{y}(\mathbf{x}')) = \hat{\sigma}^2 \mathbf{f}'(\mathbf{x}') (\mathbf{F}'\mathbf{F})^{-1} \mathbf{f}(\mathbf{x}'). \quad (5.6)$$

Yet further, an estimate $\hat{\sigma}^2$ is used in model validity checking, specifically during model diagnostics. For example, the formula for the studentized residual of the i th observation is

$$t_i = \frac{y_i - \hat{y}_i}{\hat{\sigma} \sqrt{1 - h_{ii}}} \quad (5.7)$$

where h_{ii} is the i th diagonal of the hat matrix $\mathbf{H} = \mathbf{F}(\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}'$. The studentized residual is an important quantity as it allows for the identification of outliers which may be having undue influence on the model fit.

Further, as discussed in [6, 44, 51, 68] an additional benefit to incorporating pure experimental replicates in the design is to assess the correctness *of the assumed model itself*. Consider experiments in, say, the physical, chemical, or material sciences. There may be some true physics or chemistry based model which relates the response variable to the experimental variables, denote this as

$$y = f(\mathbf{x}', \boldsymbol{\theta}) + \epsilon. \quad (5.8)$$

In some scenarios the model form may f be known, e.g., based on physics-based laws, and the objective may be, for the specific application at hand (i.e., for the specific

ranges of $\mathbf{x}$, to use the experimental data to estimate the parameter vector $\boldsymbol{\theta}$. In many applications however, the form of f may not be known and must be assumed and empirically quantified. In such a scenario it is common to approximate f with a low-order Taylor series expansion and a resulting linear model such as

$$y \approx \mathbf{f}'(\mathbf{x}')\boldsymbol{\beta} + \eta + \epsilon \quad (5.9)$$

where $\mathbf{f}'(\mathbf{x}')\boldsymbol{\beta}$ is now a linear structure in the parameter $\boldsymbol{\beta}$ (e.g., a second-order model or a third-order polynomial in the x 's), the parameter vector $\boldsymbol{\beta}$ contains the partial derivatives of f w.r.t $\mathbf{x}$ and includes any of the $\boldsymbol{\theta}$ elements that “come along for the ride” due to the Taylor expansion of f on variable $\mathbf{x}'$ in the neighborhood of $\mathbf{0}$, and the error term η represents the aggregated Taylor approximation error due to approximating true function f by linear model vector $\mathbf{f}'$. Thus, the assumption that

$$\mathbf{f}'(\mathbf{x}')\boldsymbol{\beta} \approx f(\mathbf{x}', \boldsymbol{\theta}) \text{ on } \mathcal{X} \text{ given } \boldsymbol{\theta} \in \boldsymbol{\Omega} \quad (5.10)$$

is a fundamental assumption of the modeling as well ($\boldsymbol{\Omega}$ represents the parameter space). Further, η now represents the model approximation error. If the Taylor approximation of f on $\mathcal{X}$ is ‘correct’, then $\eta = 0$. This condition can be tested via the *Lack-of-Fit* (LOF) test provided that pure experimental replicates were included in the design.

5.4.1 Recommended estimate of σ^2 in designed experiments

A number of contemporary authors in the fields of DoE and RSM are strongly recommending $\hat{\sigma}_{\text{PE}}^2 = MSPE$ (mean-square pure error) as an estimate of σ^2 [43, 51, 53, 80]. The main reason is that $\hat{\sigma}^2 = MSE$, while containing errors due to *LOF*, is a model based estimate and, therefore, is, in general, subject to statistical bias generally

and especially when the fitted model does not contain important effects [43, 53].

Gilmour and Trinca (2012) [43] thus demonstrate that inference e.g., Eq.s 5.5 and 5.6 is generally improved when the estimate is based only on pure error, and specifically it has been shown to be an unbiased estimator. Sanders and Hild (2012) [80] provide a contribution discussing graphical model diagnostics for detecting outliers in unreplicated 2^K and 2^{K-P} designs. Their paper highlights the difficulty in detecting outliers when pure replication is not included in the design. The efficiency of model diagnostics can be greatly improved if, for example the studentized residuals in Eq. 5.7 appeal to an unbiased estimate of σ^2 such as that given by pure replicates.

Montgomery and Jones (2017) address this problem in the context of screening experiments (i.e., the fitted linear model is first-order plus interaction terms and the objective is to learn which factors are most significant). These authors focus on providing a rule-of-thumb for how much partial replication to include in small two-level factorial designs. They advocate that linear model analysis in designed experiments should apply the non-biased estimator $\hat{\sigma}_{PE}^2 = MSPE$ for σ^2 . Further, they give a simple yet appealing argument regarding the effects of the df of the pure error variance estimate on the statistical power for detecting important factor main effects and interactions. For tests of size $\alpha = 0.05$ they provide a table of t -distribution critical values as a function of df . We provide this table in Table 5.6. The table shows that for $df = 1$ the effect size (i.e., a $\hat{\beta}$ must be nearly 13 times larger than its standard error to be detected and declared significant. A very substantial decrease in the t -critical value occurs when $df = 4$, and so an effect must only be 2.78 times larger than its standard error to be detected and declared significant. These authors suggest that, especially in industrial settings, practitioners should consider effect sizes of 2 standard deviations important and ensure that a design has the ability to detect effects of this size with appreciable power. They give a target power of 0.75-0.8 as a reasonable goal.

As an example they discuss the 2^4 factorial design which has $N = 16$ runs. They show that only including 4 pure replicates to give a $N = 20$ size experiment pushes the power to detect effects sizes of 2 standard deviations into this target range. From the perspective of effect size, 4 pure replicates are recommended as a minimum to include in the design. Power, of course, will be problem dependent and also depend on the intended model (e.g., number of parameters) to be fit.

The power of the t -test for regression coefficients is given in [60]. The competing hypotheses for the j th linear coefficient are $H_0 : \beta_j = 0$ vs. $H_A : \beta_j \neq 0$. The test statistic is

$$t^* = \frac{\hat{\beta}_j}{\sqrt{\widehat{\text{Var}}(\hat{\beta}_j)}} \quad (5.11)$$

when, if based on pure error, $\widehat{\text{Var}}(\hat{\beta}_j) = MSPE(\mathbf{F}'\mathbf{F})_{jj}^{-1}$ where $(\mathbf{F}'\mathbf{F})_{jj}^{-1}$ is the j th diagonal entry. The df associated with this statistic is $N - M$ as opposed to $N - P$ if the variance estimate were model based. Suppose that in truth $E(\beta_j) \neq 0$. Under H_0 , $t^* \sim t_{N-M}$. Under H_A $t^* \sim t_{N-M,\delta}$ where the non-centrality parameter

$$\delta = \frac{\beta_j}{\sigma_{PE}\sqrt{(\mathbf{F}'\mathbf{F})_{jj}^{-1}}}. \quad (5.12)$$

Eq. 5.12 illustrates that the non-centrality parameter is interpreted as the number of times larger the true β_j is than its standard deviation and that this term is influenced by the effect size β_j , the true value of σ as well as the model $\mathbf{F}$. The power g for a two-sided size α test is

$$g(\delta) = \Pr\left(|t^*| > t_{N-M}^{1-\frac{\alpha}{2}} | H_A\right). \quad (5.13)$$

Regarding LOF test and its power, we were unable to find any references discussing this directly. Thus we provide such a discussion in the next section.

5.4.2 The lack-of-fit (LOF) test.

The LOF test is constructed via a straightforward ANOVA concept where the residual error variance can be decomposed into two components: the pure-error variance and the lack-of-fit error variance. The former provides a measure of spread among pure experimental replicates, and it has been demonstrated that an estimate of this variance based on pure replicates is preferred as the estimate of the residual error variance for the linear-model calculations as it is unbiased (in a statistical sense). The decomposition of errors into the LOF components represent a pure data-driven assessment of the validity of the assumed Taylor expansion to f . Indeed, if the LOF errors are negligible (i.e., the assumed expansion of f is reasonable) the resulting error variance will be on the order of (or smaller than) the error variance of the pure replicates.

Many textbooks such as [44, 68] provide a construction of the LOF test, but do not provide the distribution of the test statistic under the alternative nor discuss power. We provide such a contribution in the appendix of this chapter. We briefly summarize the main results here. Let $\mathbf{y} : N \times 1$ be the vector of response measurements. We assume $\mathbf{y} \sim \mathcal{N}_N(\boldsymbol{\mu}, \sigma^2 \mathbf{I}_N)$. The researcher will assume a linear model as a function of covariates $\mathbf{X} : N \times K$. The model matrix is denoted $\mathbf{F} : N \times p$ has rank p with linear parameter vector $\boldsymbol{\beta} : p \times 1$. Suppose that in the assumed structure $\mathbf{F}$ there are M unique design points with N_i replicates per group. Thus $\sum_{i=1}^M N_i = N$. The scalar forms of the sum-of-square (SS) terms corresponding to model residual errors,

pure-error, and LOF errors are given in [68] as:

$$SSE = \sum_{i=1}^M \sum_{j=1}^{N_i} (y_{ij} - \hat{y}_i)^2 \quad (5.14)$$

$$SSPE = \sum_{i=1}^M \sum_{j=1}^{N_i} (y_{ij} - \bar{y}_i)^2 \quad (5.15)$$

$$SSLOF = \sum_{i=1}^M N_i (\bar{y}_i - \hat{y}_i)^2. \quad (5.16)$$

We show in the appendix that

$$SSE = SSLOF + SSPE. \quad (5.17)$$

Estimation of the variance parameters is accomplished by equating to the mean-square errors, specifically:

$$\hat{\sigma}_{\text{PE}}^2 = \frac{SSPE}{N - M} = MSPE. \quad (5.18)$$

The mean-squares attributable to LOF is

$$MSLOF = \frac{SSLOF}{M - p}. \quad (5.19)$$

The expected value of MSLOF is [68]

$$E(MSLOF) = \sigma_{\text{PE}}^2 + \frac{\sum_{i=1}^M N_i (E(y_i) - \mathbf{f}'(\mathbf{x}'_i)\boldsymbol{\beta})^2}{M - p}. \quad (5.20)$$

The null and alternative hypothesis for the LOF test are

$$H_0 : E(MSLOF) = \sigma_{PE}^2 \text{ vs.} \quad (5.21)$$

$$H_A : E(MSLOF) > \sigma_{PE}^2.$$

A test statistic for lack-of-fit of model $\mathbf{F}$ is given in [68] as

$$F = \frac{SSLOF/(M-p)}{SSPE/(N-M)} = \frac{MSLOF}{MSPE} \quad (5.22)$$

and under H_0 we have $F \sim F_{M-p, N-M}$. The distribution of F_0 under H_A was not provided in any references we reviewed. Therefore, we provide a full derivation of the LOF statistic and its distribution under both the null and alternative hypothesis in the appendix.

5.4.3 The Power of the LOF test.

In the appendix we show that the LOF statistic under the alternative is non-central F -distributed $F \sim F_{M-p, N-M, \lambda}$ with non-centrality parameter

$$\lambda = \frac{\boldsymbol{\mu}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \boldsymbol{\mu}}{\sigma^2} \quad (5.23)$$

for some general mean $E(\mathbf{y}) = \boldsymbol{\mu}$. The hat matrix $\mathbf{H} = \mathbf{F}(\mathbf{F}'\mathbf{F})^{-1}\mathbf{F}'$ in the non-centrality parameter is the result of the assumed Taylor approximation to f because the rows of the model matrix $\mathbf{F}$ are $\mathbf{f}'$ (recall Eq. 5.10). Let the general mean $\boldsymbol{\mu}$ be given by some other linear model which is more general (e.g., higher order) than the one assumed and let the resulting true model matrix be denoted by $\mathbf{F}^*$. Then we

may rewrite the non-centrality parameter as

$$\begin{aligned}\lambda &= \frac{\boldsymbol{\beta}^{*'} \mathbf{F}^{*'} \oplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F}^* \boldsymbol{\beta}^* - \boldsymbol{\beta}^{*'} \mathbf{F}^{*'} \mathbf{F} (\mathbf{F}' \mathbf{F})^{-1} \mathbf{F}' \mathbf{F}^* \boldsymbol{\beta}^*}{\sigma^2} \\ &= \frac{\boldsymbol{\beta}^{*'} \mathbf{F}^{*'} \mathbf{F}^* \boldsymbol{\beta}^* - \boldsymbol{\beta}^{*'} \mathbf{F}^{*'} \mathbf{F} (\mathbf{F}' \mathbf{F})^{-1} \mathbf{F}' \mathbf{F}^* \boldsymbol{\beta}^*}{\sigma^2}\end{aligned}\quad (5.24)$$

Eq. 5.24 shows that the numerator of the non-centrality parameter is an aggregation of squared errors attributable to model mis-specification. If the model is not mis-specified then the numerator should be near 0 (this is shown in the appendix). When the the model is highly mis-specified the numerator will be much larger than the residual error variance in the denominator, and it is more likely to generate an observed F -statistic that is not consistent with the null hypothesis. Unfortunately, producing a more informative formula for λ which clearly illustrates how it aggregates LOF errors, and, exactly what effects the LOF is testing for, can only be accomplished by examining specific situations with a true model that has specified form of $\mathbf{F}^*$. See Draper and Herzberg (1971) [35] for such a discussion.

The non-centrality parameter governs the location and shape of the non-central F distribution. The power of the LOF test is viewed as a function of λ , and we think of λ as the number of times larger the sum-of-squared model mis-specification errors is compared to the residual error variance. For a size α test, the power g is computed as

$$g(\lambda) = \Pr \left(F \geq F_{(M-p, N-M)}^{1-\alpha} | H_A \right) \quad (5.25)$$

where $F_{(M-p, N-M)}^{1-\alpha}$ is the $100 \times (1 - \alpha)$ percentile of the $F_{(M-p, N-M)}$ distribution.

In applications (especially in physics, chemistry, and material science) it is not uncommon that the degrees-of-freedom terms $\nu_1 = df_{\text{LOF}} = M - p$ and $\nu_2 = df_{\text{PE}} = N - M$ are both small. This is because the total sample size N is usually cost constrained and

some number of these will be allocated to pure experimental replicates. Note here the interpretation of $\nu_2 = df_{\text{PE}} = N - M$: this quantity is the number of distinct points in the design which are replicated (M is the number of unique points in the design). Further, the interpretation of $\nu_1 = df_{\text{LOF}} = M - p$ is the number of additional linear terms that could be added to the model in an effort to mitigate a LOF. Thus, the researcher would, potentially, like this quantity to be large if possible and if the true model is unknown. Looking at power in this situation will help the researcher understand the trade-offs of increasing/decreasing the number of replicates (e.g., decreasing/increasing M). To illustrate these tradeoffs, we compute a set of power curves for different replication scenarios which are motivated from a common classical experimental designs. Consider a $K = 3$ factor design scenario where one can afford to run in total $N = 16$ independent experiments. Suppose that the researcher is intending to fit the second order model which has $p = 10$ parameters. A minimum of 10 distinct points are required to fit this model, and so the researcher can consider a number of replication scenarios. We list some example replication scenarios in Table 5.7. Note that the described scenarios are not exhaustive, but the number of scenarios shown in regard to ν_1 and ν_2 is exhaustive in that we have covered all possible degree-of-freedom decomposition scenarios for the LOF and PE. With respect to allocation of the replicates from the N experimental points from the perspective of power of the LOF test, it is clear from Table 5.7 that we need not consider degree-of-freedom scenarios 1 and 7 (as one of the ν terms is 0 and hence the LOF test cannot be conducted). A common recommendation for replication in this situation is the classical *central-composite design* (CCD) [68]. The CCD for this scenario is comprised of $2^K = 8$ factorial (vertex) points, $2K = 6$ axial (face center) points, and $N_C = 2$ replicates at $(0, 0, 0)$ (the center of $\mathcal{X}$). An illustration of this design is shown in Figure 5.10.

We provide power curves for the df scenarios of Table 5.7 in Figure 5.11. For these low df scenarios, it is apparent that appreciable power is achieved for large λ . The figure also illustrates that, for fixed λ , it is better (in terms of having larger power) to have more df in the PE term (denominator) than in the LOF term (numerator). Note that these differences in power are large, e.g., spanning a factor of 10 between the largest and smallest power scenario when $\lambda = 5$. The purple curve shows the power of the LOF test for df scenario 6, i.e., the CCD shown in Figure 5.10, which has $\nu_1 = 1$ df for the LOF term and $\nu_2 = 5$ df for the PE term. This shows that, while this scenario offers the best ability to fix the model in case of detected LOF (i.e., $\nu_2 = 5$), even for very large $\lambda = 25$ the power is still close to the significance level $\alpha = 0.05$. Indeed, the commonly recommended textbook design, the CCD, has very poor power for detecting LOF.

The result of the power analysis provides an interesting result: For small df scenarios (which are common in practice):

1. From the perspective of detecting LOF, it is more beneficial to have more pure replication so as to increase the df of the PE term;
2. From the perspective providing a larger number of opportunities to mitigate LOF, it is more beneficial to allocate more df to the LOF term;
3. Considerations 1. and 2. are in competition to each other for small to moderate N and M .

5.5 Using PSO to generate IV -optimal designs with a specified replication structure.

As mentioned in the introduction, Jensen (2018) has highlighted that the contemporary popularity of optimal design has overshadowed and downplayed the importance of planning for and including pure replication in designed experiments.

This is because replication structures are not easily ‘built in’ to the optimality criterion or the search algorithms. For example, when applying optimal design to a discretization of $\mathcal{X}$ via a set of candidate points, a point-exchange algorithm is applied to search for the optimal design. There are two choices: (1) include specified replicates of each point in the candidate set in the hopes that the objective will favor a replication structure (which doesn’t generally happen), or (2) choose specific points that must be included as replicates and use the optimal design search to augment these points. Some optimality criteria for specific design scenarios naturally generate some replicate points in a design (if the criteria favors it), but this is not generally the case. Thus, algorithms to generate optimal designs with a specified replication structure is still something of an open problem. In this section we propose an approach to this problem which may be applied for exact designs provided that N is not so large that the number of replication scenarios for a given K cannot be studied exhaustively.

5.5.1 User specified replication structure in the PSO generating algorithm

We are able to allow the user to request and enforce a replications structure in the optimal design search by altering the definition of the IV -criterion (i.e., the optimality objective). Let

$$IV_{\text{rep}}(\mathbf{X}^*, \mathbf{R}) = \frac{1}{V} \text{tr} \left\{ (\mathbf{F}^{*'} \mathbf{R} \mathbf{F}^*)^{-1} \mathbf{W} \right\} \quad (5.26)$$

be the IV -criterion which defines the integrated variance scale for a specified replication structure and

- $\mathbf{X}^* : M \times K$ is the design matrix for K factors with a maximum of M unique design points,
- $\mathbf{F}^* : M \times p$ is the corresponding model matrix, and

- $\mathbf{R} : M \times M = \text{diag} \{N_i\}$ where $\sum_{i=1}^M N_i = N$ is the user-specified replication matrix indicating how many times each of the M unique designed points are replicated.

Note that this formulation of the model information matrix is equivalent to the information matrix with all design points (including replicates) included, that is,

$$\mathbf{F}^{*'} \mathbf{R} \mathbf{F}^* = \mathbf{F}' \mathbf{F}.$$

We are currently unaware of other algorithms that enable a search for the exact optimal design that allows a user specified replication structure but that does not require additional specification of the specific points (i.e., the center point) that must be replicated. The approach of simply redefining the objective as in Eq. 5.26 works seamlessly with our PSO optimal design search code. That is, we only had to program new objective functions and did not have to alter the PSO algorithm in any way. This seems like a nice advancement to the capability of searching for optimal designs.

Now, similar to how an optimality criterion may naturally support a replication structure, this definition of the optimality criterion can only allow the user to specify the *maximum number of unique points* M . It is possible that a specified replication structure may ‘reduce’ to a replication structure that is more preferred by the optimality criterion. For example, say we are searching for a $N = 12$ point design and the user specifies that $M = 10$ points should be unique and that the 9th and 10th point should each be replicated twice. If it is true that the optimality criterion prefers $M = 9$ unique points with the 9th point replicated 4 times, then this design will be returned by the search.

5.5.2 An exhaustive study of some small K moderate N cases

A proposed approach allows the user to impose a replication structure during the PSO search for the IV -optimal design. Provided that N is not so large so as to create an unwieldy large number of replication scenarios, we can use PSO to exhaustively study and compare optimal designs with different replication structures for a fixed $\{K, N\}$ pair. To illustrate our approach we implement the search on all replication scenarios for the $\{K, N\}$ pairs shown in Table 5.8. The number of replication scenarios is also listed (a PSO search was conducted for each scenario). It is important to attempt to generate optimal design searchers for each scenario because these cover different states of the numerator and denominator degrees-of-freedom for the LOF test. For each scenario we conducted $n_{\text{run}} = 210$ PSO searches using $S = 250$ particles.

In the following subsections we provide a table of the replication scenarios, IV -scores for the optimal model found for each search, and some deeper analysis into some select models which include replicates.

5.5.2.1 $K = 2, N = 10$ IV -optimal designs with replication The full list of the 12 possible replication scenarios are shown in Table 5.11 column ‘rep.vector.set’. We ran $n_{\text{run}} = 210$ PSO searches using $S = 250$ particles for each scenario, while applying the optimality objective shown in Eq. 5.26 using the appropriate $\mathbf{R}$ matrix. The table also lists the space of ν_1, ν_2 , that is, the df scenarios for the LOF test. In the last two columns the IV -score and IV -relative efficiency (relative to the best design found in this set) are shown. Not surprisingly the first scenario (no specified replication) has the lowest IV -score. In fact, for the K, N scenario, the IV -optimal model is a CCD with two replications of the center point $(0, 0)$. Thus, the scenario 2 model, which specifies that two replicates must be included, also produced the CCD. There were four other scenarios which had IV -releff greater than 90%: scenarios 3 through 6. As

we have demonstrated that the LOF test is more powerful when the denominator df is larger, perhaps these models are reasonable choices for the practitioner as they are trading a bit of IV -efficiency for a better ability to detect LOF.

We analyze further the properties of the four designs with the highest IV -scores: scenarios 1 (naturally generated one point replicated twice), 3 (one point replicated thrice), 4 (two points each replicated twice), 6 (one point replicated twice, one point replicated thrice). We first present the generated designs in Figure 5.12. Scenario 1 shows the $K = 2$ factor CCD with two center replicates. Scenario 3 requested the search to look for designs that replicated a single point thrice, and the result was the removal of one edge center point, the three replicates moved toward that location off center, and the two edge center points with $x_1 = \pm 1$ moved slightly negative. Similar balancing acts can be seen happening in the scenario 4 design (two points each replicated twice) and the scenario 6 design (two points, one replicated twice, the other thrice). The latter two designs give the researcher an opportunity to assess the constant variance assumption at two distinct locations in $\mathcal{X}$. We provide the FDS plots comparing the four designs in Figure 5.13. While the CCD is IV -optimal, the designs with other replication structure have the lower 50% of SPVs slightly below the IV -optimal design, while the upper 40% of the SPVs are above that of the CCD , albeit this is not a large increase. To further illustrate the properties of these designs Figure 5.14 contains SPV surface plots of these scenarios. These figures indicate that designs from replication scenarios 3 and 4 take a large hit on SPV at the $x_2 = 1$ boundary, while the design from scenario 6 takes a hit on both $x_2 = \pm 1$ boundaries. Note that, however, the designs from scenarios 3, 4, and 6 have large portions of the interior of $\mathcal{X}$ with smaller SPV than the CCD .

5.5.2.2 $K = 2$, $N = 12$ IV -optimal designs with replication The full list of the 12 possible replication scenarios are shown in Table 5.12 column ‘rep.vector.set’. We ran $n_{\text{run}} = 210$ PSO searches using $S = 250$ particles for each scenario, while applying the objective shown in Eq. 5.26 using the appropriate $\mathbf{R}$ matrix. The table also lists the space of ν_1, ν_2 , that is, the df scenarios for the LOF test. Note that this set of df scenarios for the LOF test is the same as that represented in the power curves of shown in Figure 5.11. In the last two columns the IV -score and IV -relative efficiency (relative to the best design found in this set) are shown.

The results indicate that scenarios 1 through 5 generated an IV -PSO design with the same IV -score. It was confirmed that these scenarios prefer the same design, a CCD with 4 center points. As we have illustrated that larger df for the pure error term results in larger power, we will select the highest IV -efficient designs from the other df scenarios among this set for comparison. These are: scenario 6 (IV -releff = 98.97), scenario 10 (IV -releff = 95.33), scenario 14.1 (IV -releff = 92.66). Each scenario represents increased power to detect LOF. Plots of the IV -PSO designs for these scenarios are shown in Figure 5.15 with the replicated points indicated in red. These designs are further analyzed via the FDS plot which is included in Figure 5.16 each alternate replication design can be seen to have slightly worse SPV profiles than the CCD in the FDS plot. Contour plots of the resulting SPV surfaces for each design are included in Figure 5.17.

A primary result from this section that may be useful for practitioners that are confronted with small/moderate N design problems on $K = 2$ factors (either in this phase of designing a sequence of designs, or by total cost allowed for experimentation) is the following:

1. For this K, N scenario, the maximum available power for detecting LOF is limited.

2. If the researchers are very confident that they understand the correct form of f , i.e., the second-order model, then the CCD with 4 center points is *IV*-optimal and will give the smallest prediction variance on average on $\mathcal{X}$. Therefore is a reasonable choice for implementation. The power curve associated with the LOF test is the green curve shown in Figure 5.11, $\nu_2 = 3$ for the PE term.
3. If, however, knowledge of the correct f is limited, the model shown in Scenario 14.1 may be a better choice. This design offers $\nu_2 = 4$ for the PE term, and therefore, the best power. The power curve associated with this model is the red curve in Figure 5.11. Additional benefits of this design include: (1) a better (in terms of both bias and df and therefore estimation uncertainty) estimate of σ_{PE} , which is the preferred term to be used to estimate uncertainty on the model coefficients and predictions because it is model independent, (2) three distinct opportunities to assess the homoscedasticity assumption of the linear modeling, and (3) more robust outlier detection as this estimate can be used in the standardized residual formula and can therefore assess the possibility that other design points may be outliers (which is not the case without replication).

5.5.2.3 $K = 3$, $N = 14$ *IV*-optimal designs with replication The full list of the 12 possible replication scenarios are shown in Table 5.13 column ‘rep.vector.set’. We ran $n_{\text{run}} = 210$ PSO searches using $S = 250$ particles for each scenario, while applying the objective shown in Eq. 5.26 using the appropriate $\mathbf{R}$ matrix. The table also lists the space of ν_1, ν_2 , that is, the df scenarios for the LOF test. In the last two columns the *IV*-score and *IV*-relative efficiency (relative to the best design found in this set) are shown. Similar to the previous sections, we analyze further the model with the highest *IV*-score, as well as the next 3 highest *IV*-efficient models from the other df scenarios (excluding the $df = 0$ scenarios). For this set we will discuss and compare

designs from replication scenarios 1 ($IV\text{-releff} = 100$), scenario 4 ($IV\text{-releff} = 97.54$), scenario 5 ($IV\text{-releff} = 96.49$), and scenario 7 ($IV\text{-releff} = 97.2$).

We present the competing designs for these scenarios in Figure 5.18. We analyze the models via FDS in Figure 5.19. For this K there appears to be very little difference in FDS profiles. Therefore the designer may wish to select that design which has largest power for detection of LOF.

5.5.2.4 $K = 3$, $N = 16$ IV -optimal designs with replication The full list of the 12 possible replication scenarios are shown in Table 5.14 column ‘rep.vector.set’. We ran $n_{\text{run}} = 210$ PSO searches using $S = 250$ particles for each scenario, while applying the objective shown in Eq. 5.26 using the appropriate $\mathbf{R}$ matrix. The table also lists the space of ν_1, ν_2 , that is, the df scenarios for the LOF test. Note that this set of df scenarios for the LOF test is the same as that represented in the power curves shown in Figure 5.11. In the last two columns the IV -score and IV -relative efficiency (relative to the best design found in this set) are shown. Similar to the previous sections, we further analyze the model with the highest IV -score, as well as the next 3 highest IV -efficient models from the other df scenarios (excluding the $df = 0$ scenarios). For this set we will discuss and compare designs from replication scenarios 1 ($IV\text{-releff} = 100$), scenario 5 ($IV\text{-releff} = 98.2$), scenario 11 ($IV\text{-releff} = 94.74$), and scenario 15 ($IV\text{-releff} = 91.91$).

We present the competing designs for these scenarios in Figure 5.20. We analyze the models via FDS in Figure 5.21. For this K there appears to be very little difference in FDS profiles. Therefore, the designer may wish to select that design which has largest power for detection of LOF.

5.5.2.5 $K = 4$, $N = 20$ IV -optimal designs with replication This design scenario was explicitly mentioned in Jensen (2018) [51] in the context of identifying the ‘best’

use of replication. The full list of the 18 possible replication scenarios are shown in Table 5.15 column ‘rep.vector.set’. We ran $n_{\text{run}} = 210$ PSO searches using $S = 250$ particles for each scenario, while applying the objective shown in Eq. 5.26 using the appropriate $\mathbf{R}$ matrix. The table also lists the space of ν_1, ν_2 , that is, the df scenarios for the LOF test. In the last two columns the IV -score and IV -relative efficiency (relative to the best design found in this set) are shown. Note that designs from scenarios 1-3 all have $IV\text{-releff} = 100$. We confirmed that these designs were the same. It turns out that the IV -criterion prefers one point replicated 3 times, thus giving 2 df for pure error. There are many designs in this set with $IV\text{-releff} > 90\%$. We will compare the properties of the IV -optimal design to the design in scenario 11 which has a high $IV\text{-releff}$ (94.23) and is attractive in that it gives the recommended minimum df for pure error and has 4 distinct points in $\mathcal{X}$ replicated thereby giving an opportunity to get some assessment of the homoscedasticity assumption.

We analyze the models via FDS in Figure 5.22. For this K there appears to be very little difference in FDS profiles between the design of scenario 1 (one point replicated 2 times) and that of scenario 11 (4 points replicated twice each). Therefore the designer may wish to select that design which has largest power for detection of LOF. We present the design for scenario 1 in Table 5.9 and the design for scenario 11 in Table 5.10.

5.5.3 Discussion

The benefit of using pure error replication to estimate σ^2 in the context of linear model analysis has long been known. The discussion regarding the use of pure error replicates for estimating σ^2 goes back at least to the 1950s with Cox (1958) [32]. The primary benefit is that, in contrast to the MSE (i.e., combined LOF and PE estimate), $\hat{\sigma}_{\text{PE}}^2$ is the only estimate that is guaranteed to be statistically unbiased.

This is because the MSE is also aggregating errors due to the approximation of the true unknown model. The benefits of performing estimation of σ^2 using PE replicates impacts all aspects of the linear model analysis, including uncertainty estimates of the regression coefficients, propagation of uncertainty into predicted values, model diagnostics and outlier identification, and assessing model LOF.

More contemporary authors such as Gilmour and Trinca (2012), Jones and Montgomery (2017), Sanders and Hild (2012), and Jensen (2018) have renewed this discussion in the context of optimal DoE [43,51,53,80]. They all highlight that optimal DoE has gained contemporary popularity over classical designs due to advancements in computing and the fact that optimal DoE can “design for the experiment” by incorporating a fixed affordable N as part of the design formulation problem. Because of this optimal DoE, in some senses, specifically on the scale of a specified optimality criterion, is able to produce designs with similar or better optimality properties at less cost. These authors emphasize that design optimality criterion do not explicitly incorporate or guarantee that the generated optimal design will include any number of replicates (some may naturally produce replicates though this is problem dependent and is currently not controllable).

The use of PSO as the optimization algorithm has made it possible for us to develop an algorithm which can generate exact optimal experimental designs with guaranteed replicates. This was accomplished using an adjustment to the design information matrix which includes a matrix that encodes a replication structure, as we illustrated in the IV_{rep} criterion shown in Eq. 5.26. While we only worked with the IV -criterion in this chapter, our approach may be applied to any optimality criterion which uses the design information matrix. In the studies shown for $K = 2, 3, 4$ design factors we illustrated that the user is not guaranteed to receive the requested information structure if the criterion prefers another replication structure

which subsumes the requested one. Specifically, the user is effectively specifying the maximum number of unique points M . However, this is not a problem as the primary concern, is, for fixed N , to maximize the df of the pure error term. That is, the algorithm in this sense is guaranteed to return the user with *at least* the requested df of the PE term. This is, of course, at the cost of df of the LOF term and the ability to add unanticipated terms to the model, but it is more beneficial for the totality of the LM analysis to have larger df for pure error. For all the scenarios studied we are able to make the following common observations:

1. The design with the highest IV -criterion is always that design with unspecified replication structure (though this will sometimes naturally include replicates and subsume some cases of replication for fixed N and fixed model).
2. The IV -efficiency generally decreases as df of the PE increases.
3. For all cases studied we were able to generate a design for the largest df of PE scenario with at least 90% efficiency of the best model.
4. Further analysis of these models via FDS plots showed that the hit to SPV is generally very small of the best model's SPV.
5. Given the previous observation we can recommend that it is more beneficial to produce a more highly precise estimate of σ^2 (i.e. maximize PE df) at the cost of $\sim 10\%$ IV -efficiency than it is to maximize the IV -efficiency.

Jones and Montgomery (2017) provided a simple analysis that recommends a minimum of 4 df for PE as this allows the detection effects that are 2.78 standard deviations at the $\alpha = 5\%$ level. In their paper they also provided examples for how the design impacts power of the linear model t -tests [53]. We were unable to find similar references regarding the power of the LOF test and so provided the full derivation

of the LOF test statistic under the alternative, similar to [35], and showed a simple common scenario for fixed N where LOF and PE are competing for df . This analysis illustrated that the LOF test has negligible power for 1 df PE, and at an aggregated square-error for model misspecification of 5 times the pure error variance, a df LOF of 1 and df PE of 5 only has power of 0.43 for detecting the mis-specified model. This illustrates that, if the practitioner is very uncertain regarding the true form of the model they are probably better served to plan more pure replication in the initial design phases in order to maximize the chances of identifying a more true model form.

5.6 Conclusions and Future Work

In the first part of this chapter we further validated our PSO Julia code by reproducing the results of Borkowski (2003) for all design scenarios covering $K = 1, 2, 3$ factors.

Next, we used PSO to generate $K = 4$ factor *IV*-optimal models for those cases studied in Chapter 4 and by previous authors Rodriguez et al. (2010) and Hernandez and Nachtsheim (2018) [47, 78]. This enabled a comparison of the properties of the *G*-PSO designs to *IV*-PSO designs. This comparison is important to benchmark because both criteria are choices for generating designs when the primary goal is to attain the best prediction uncertainties possible. The result of the analysis is consistent with this comparison on lower dimensions: *IV*-optimal models have smaller SPV over a majority of $\mathcal{X}$ except at the boundaries, where the *G*-optimal models are superior [78].

Last, we developed a new, and to our knowledge, first algorithm which is guaranteed to generate replicates in the produced optimal design and guarantees a specified df of PE. We illustrated our approach on numerous $K = 2, 3, 4$ factor scenarios using the *IV*-criterion and were able to provide some general recommendations for the designer: for cases where N and p are such that the number of possible replication

scenarios is so large than an exhaustive search cannot be conducted (as we did for each scenario studied in this chapter), the practitioner can focus on replication scenarios that ensure the df of the PE term is sufficient to be highly beneficial to the linear model inference, and, they are highly likely to generate designs with high optimality criterion efficiency and good SPV properties.

Future work includes applying the algorithm to other optimality criterion and model support scenarios in order to further illustrate the efficiency and efficacy of our approach.

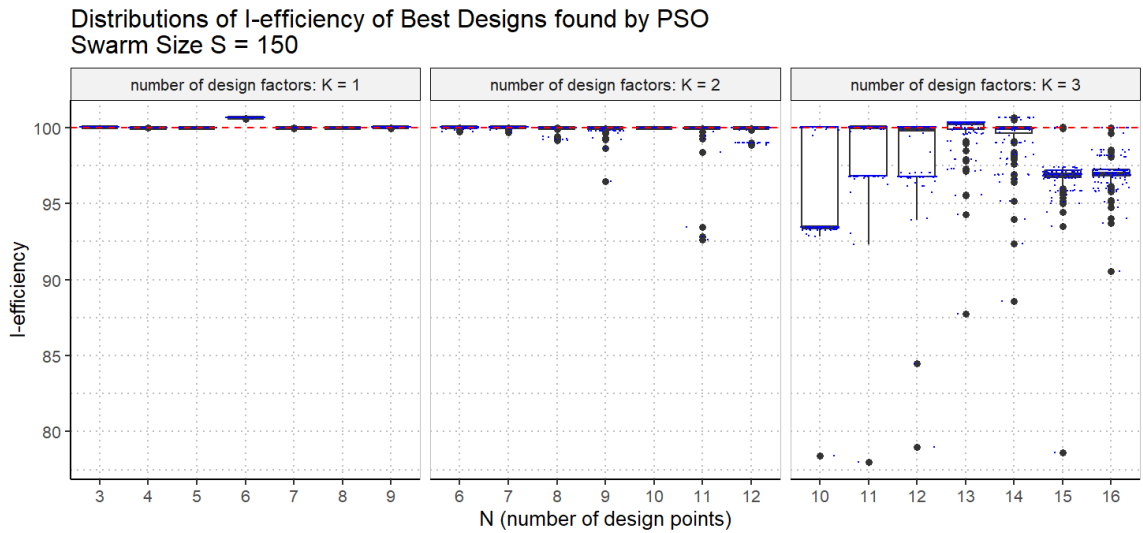
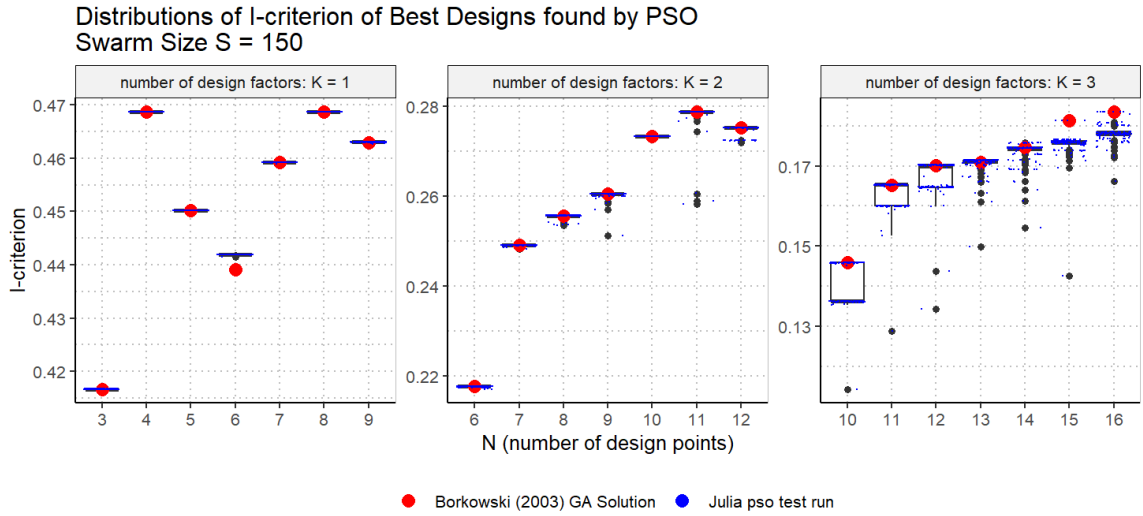


Figure 5.1: The top set of panels shows boxplots of the $n_{\text{run}} = 140$ IV -scores of the designs generated by PSO presented on the scale of Borkowski (2003) (and so larger IV -score on this scale represents a better design). The red dot represents the IV -score of the GA generated design from Borkowski (2003). The bottom set of panels show boxplots of the IV -efficiencies of the PSO generated designs with the IV -score of the GA design used in the denominator of the efficiency calculation.

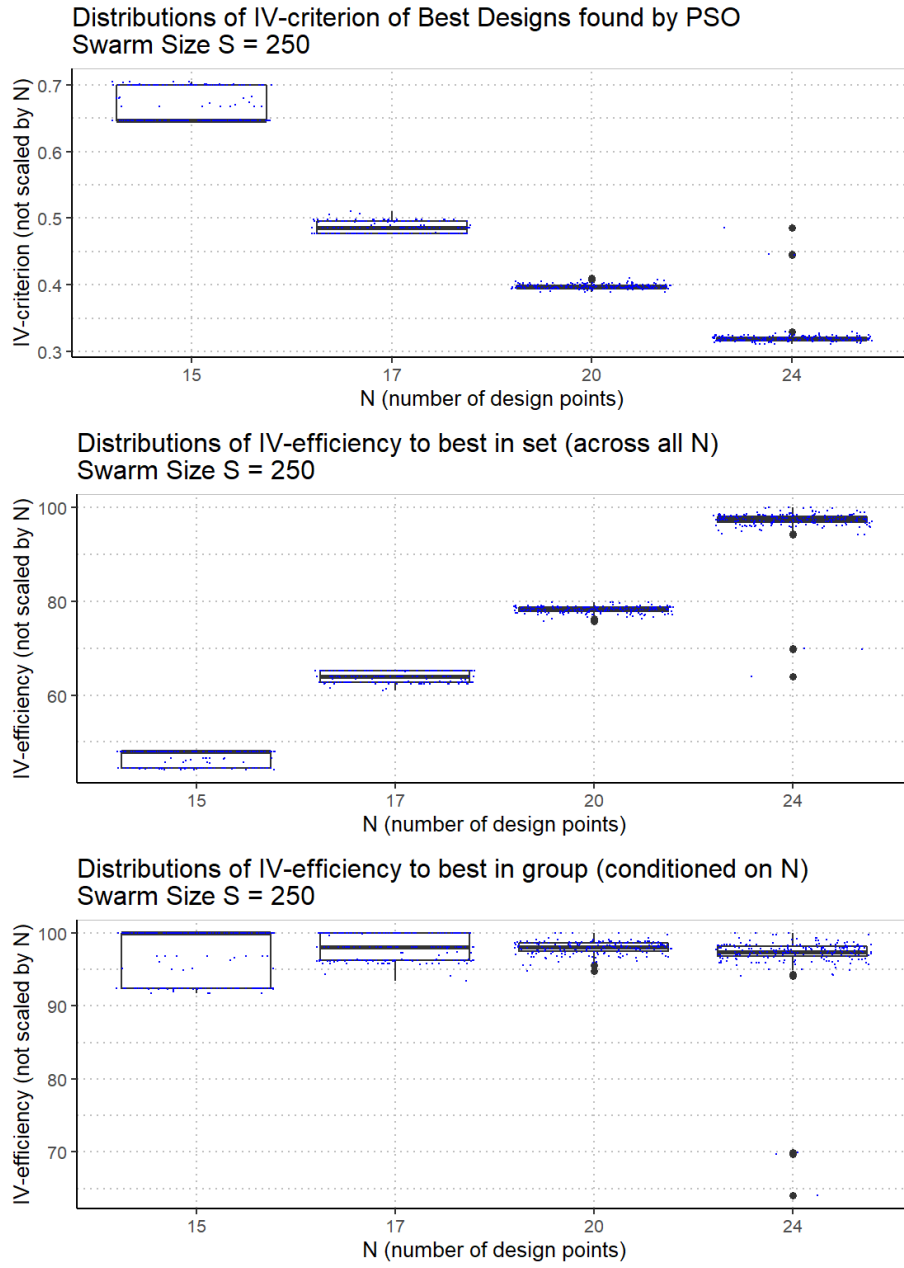


Figure 5.2: The top panel shows boxplots of the $n_{\text{run}} = 250$ IV -scores of the $K = 4$ designs generated by PSO using the unscaled IV -criterion (and so smaller score indicates a better design). The middle panel shows the IV -efficiency computed across all N . The bottom panel shows the IV -efficiency computed conditioned on N .

Table 5.1: The $K = 4$, $N = 15$ IV -PSO generated design.
$$IV = 0.6471$$

x_1	x_2	x_3	x_4
1	-1	-1	-1
-1	-0.40893	-0.90104	-1
-0.20353	0.203207	0.063367	1
-1	-0.47391	1	0.47351
0.287296	-0.28654	-1	0.286643
0.472651	-0.47432	1	-1
-1	0.999999	0.582394	-1
0.407855	1	-0.90244	-1
-1	1	-0.90138	0.408493
1	1	-1	1
0.4731	1	1	0.474138
-1	-1	-1	1
1	0.204249	0.064066	-0.20455
1	-1	1	1
-0.20327	-1	0.063248	-0.20277

Table 5.2: The $K = 4$, $N = 17$ IV -PSO generated design.

$IV = 0.4766$			
x_1	x_2	x_3	x_4
1	1	1	0.000838
-1	-1	1	-0.43831
-0.43835	-1	-1	-1
-1	-1	-0.43821	1
0.000148	-0.12164	-0.00027	-0.00025
-0.00105	1	-1	1
-1	0.727843	1	1
-1	1	0.000719	-1
-0.00043	1	0.000529	0.000555
0.000683	0.225443	1	-1
1	-1	0.438338	-1
1	0.729654	-1	-1
0.000286	-0.12193	-0.00049	-0.00034
-1	0.225282	-1	-0.00064
1	0.22552	-0.00052	1
0.438477	-1	1	1
1	-1	-1	0.438237

Table 5.3: The $K = 4$, $N = 20$ IV -PSO generated design.

$IV = 0.3894$			
x_1	x_2	x_3	x_4
-0.08584	-0.08418	-0.0118	-0.01107
1	-0.09116	0.01064	0.01075
1	-1	1	-0.406
-1	-1	0.43493	-1
0.2264	0.22328	1	-1
-0.09033	1	0.01316	0.0131
-0.07677	-1	1	1
-1	-1	-1	0.43518
1	1	0.08441	-1
0.22648	0.22267	-1	1
1	1	1	1
1	-1	-0.4057	1
-0.08576	-0.08388	-0.0114	-0.0113
1	1	-1	0.08415
-1	-0.07572	1	1
-1	0.57449	-1	-1
-1	1	-0.4093	1
0.57352	-1	-1	-1
-0.08481	-0.08389	-0.011	-0.01105
-1	1	1	-0.40972

Table 5.4: The $K = 4$, $N = 24$ IV -PSO generated design.

$IV = 0.3108$			
x_1	x_2	x_3	x_4
-1	-1	-0.093	0.1865
1	1	1	-1
1	-1	-0.0954	-1
1	1	-1	0.1474
-0.0929	-1	-1	0.1882
0.0261	0.0327	0.02481	-0.058
1	-0.0499	1	0.981
-1	-0.0739	1	-1
-0.1211	1	1	1
-1	-1	-1	-1
-0.0951	-1	1	-1
-1	1	-0.1148	-1
-0.1075	-0.2705	-0.1071	1
-1	-1	1	1
0.0208	0.0313	0.02348	-0.058
-1	0.5888	-1	1
0.0243	0.031	0.02418	-0.058
-1	1	1	0.147
1	1	-0.1219	1
1	-1	1	0.1082
-0.1174	1	-1	-1
0.0235	0.0319	0.02597	-0.057
1	-1	-1	1
1	-0.0734	-1	-1

Table 5.5: Comparison of the G and IV scores and relative efficiencies of the G -PSO and IV -PSO $K = 4$ factor designs.

N	design criteria	score		relative efficiency	
		$IV(\mathbf{X})$	$G_{\text{eff}}(\mathbf{X})$	IV -releff	G -releff
15	IV	0.6471	37.3607	100.00	52.56
	G	0.9004	71.0864	71.87	100.00
17	IV	0.4766	52.8717	100.00	71.54
	G	0.7776	73.9012	61.29	100.00
20	IV	0.3894	52.9415	100.00	66.01
	G	0.6094	80.1998	63.90	100.00
24	IV	0.3108	50.5522	100.00	58.82
	G	0.4446	85.9480	69.91	100.00

Table 5.6: t critical values for a size $\alpha = 0.05$ test as a function of df .

df	t critical value
1	12.71
2	4.30
3	3.18
4	2.78
5	2.57
6	2.45
7	2.36
8	2.31

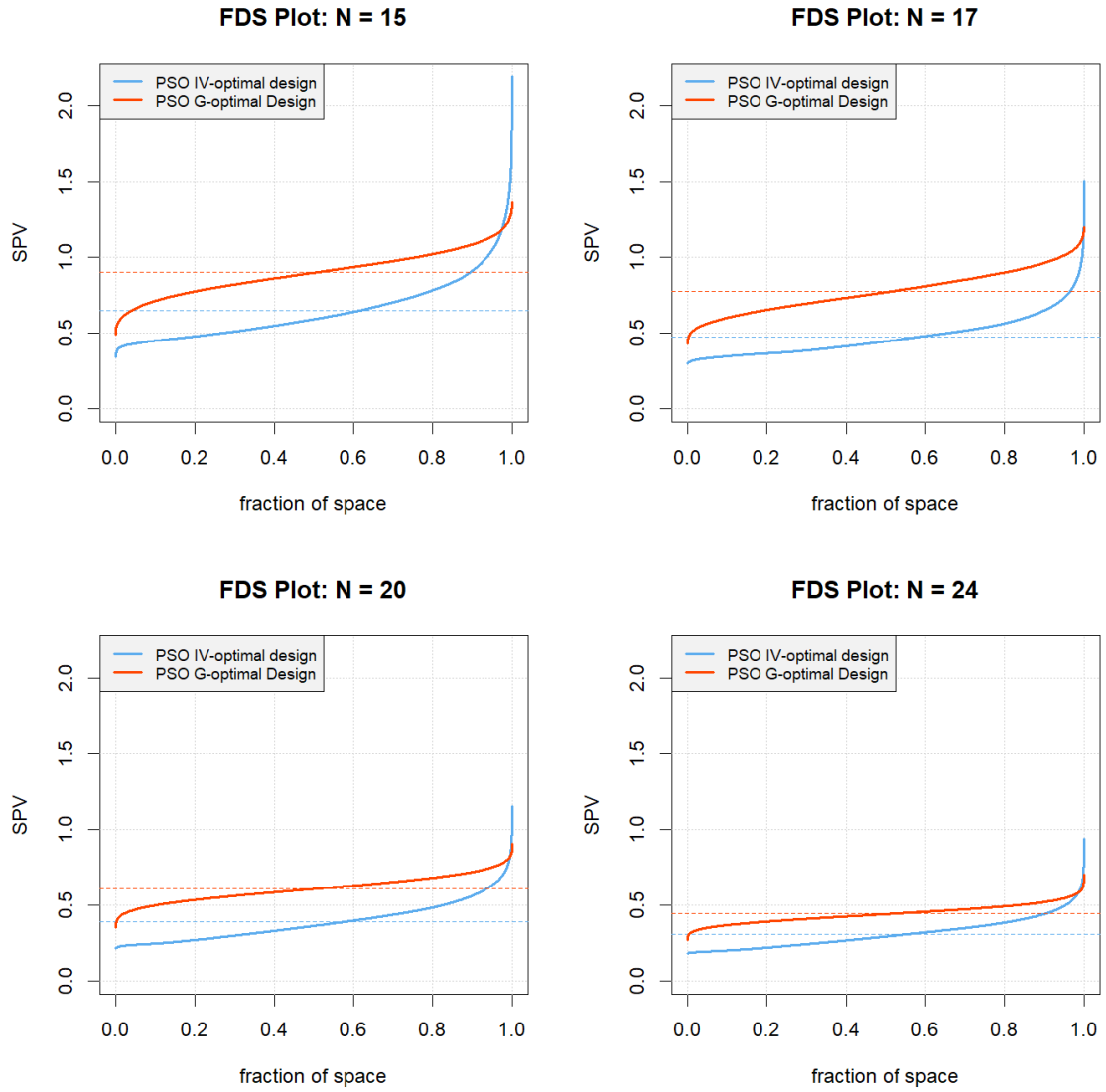


Figure 5.3: FDS plots for each $K = 4$, N comparing the G -PSO to IV -PSO design. The horizontal dotted lines show the IV -scores of each design.

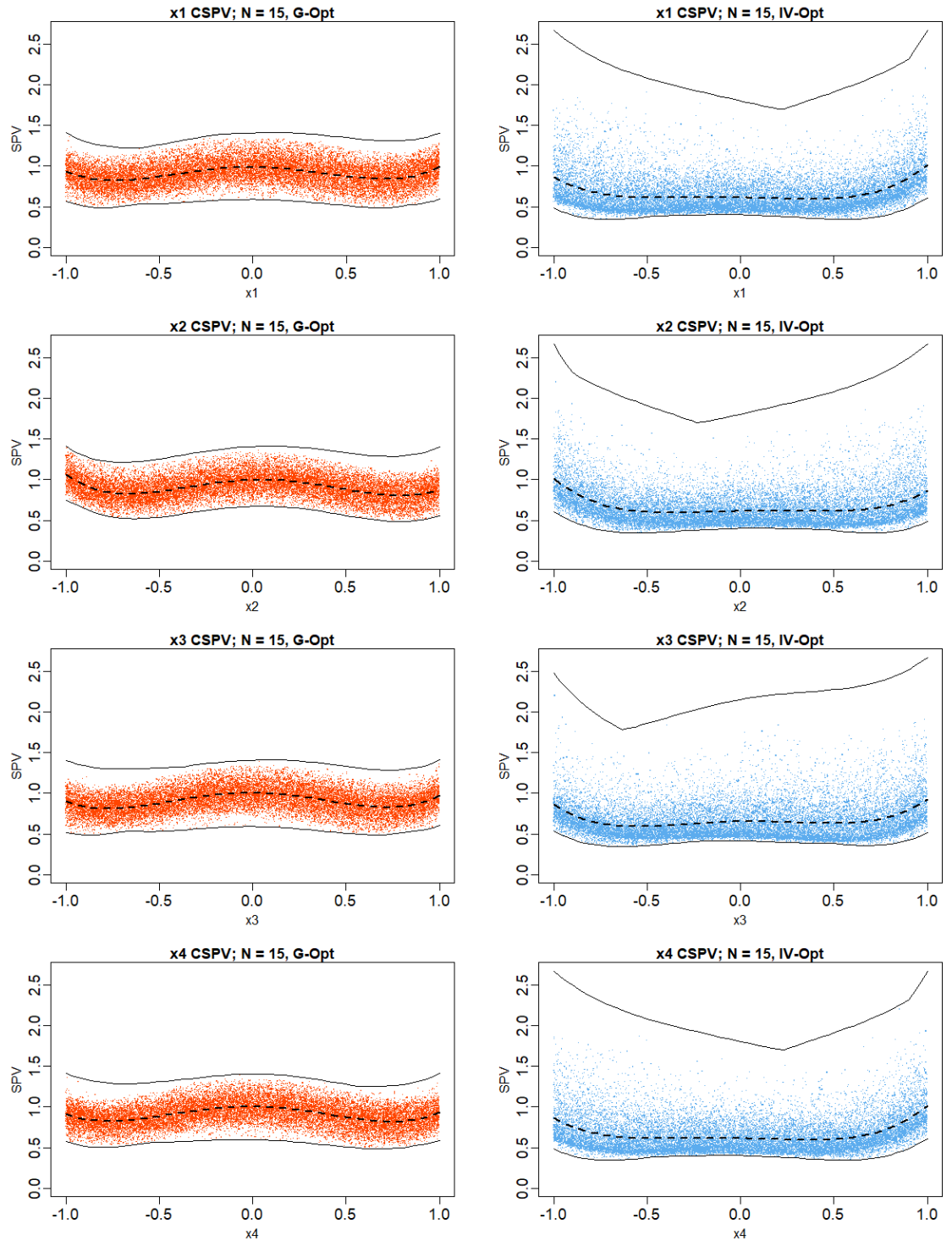


Figure 5.4: CSPV plots for $K = 4$, $N = 15$ comparing the G -PSO to IV -PSO design.

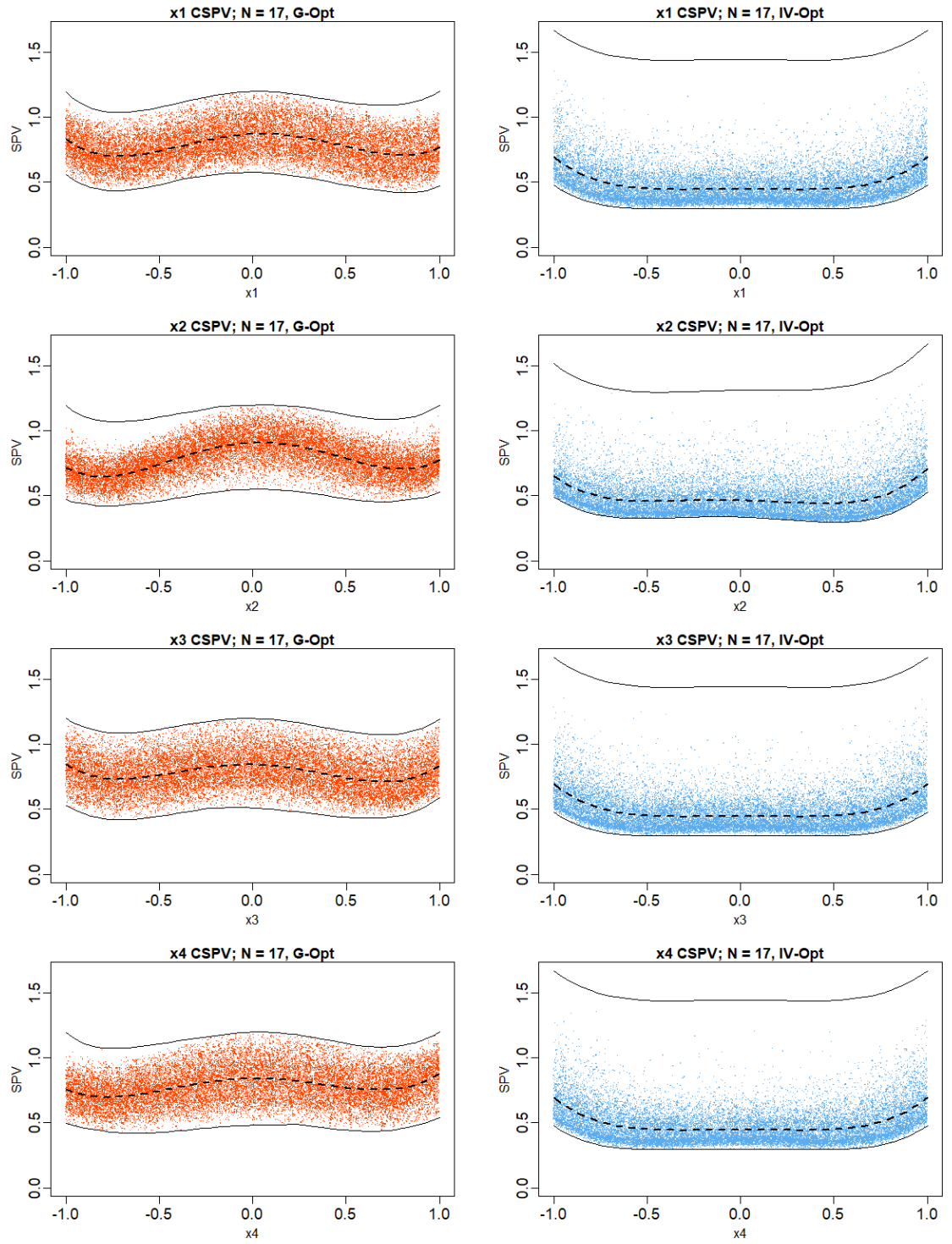


Figure 5.5: CSPV plots for $K = 4$, $N = 17$ comparing the G -PSO to IV -PSO design.

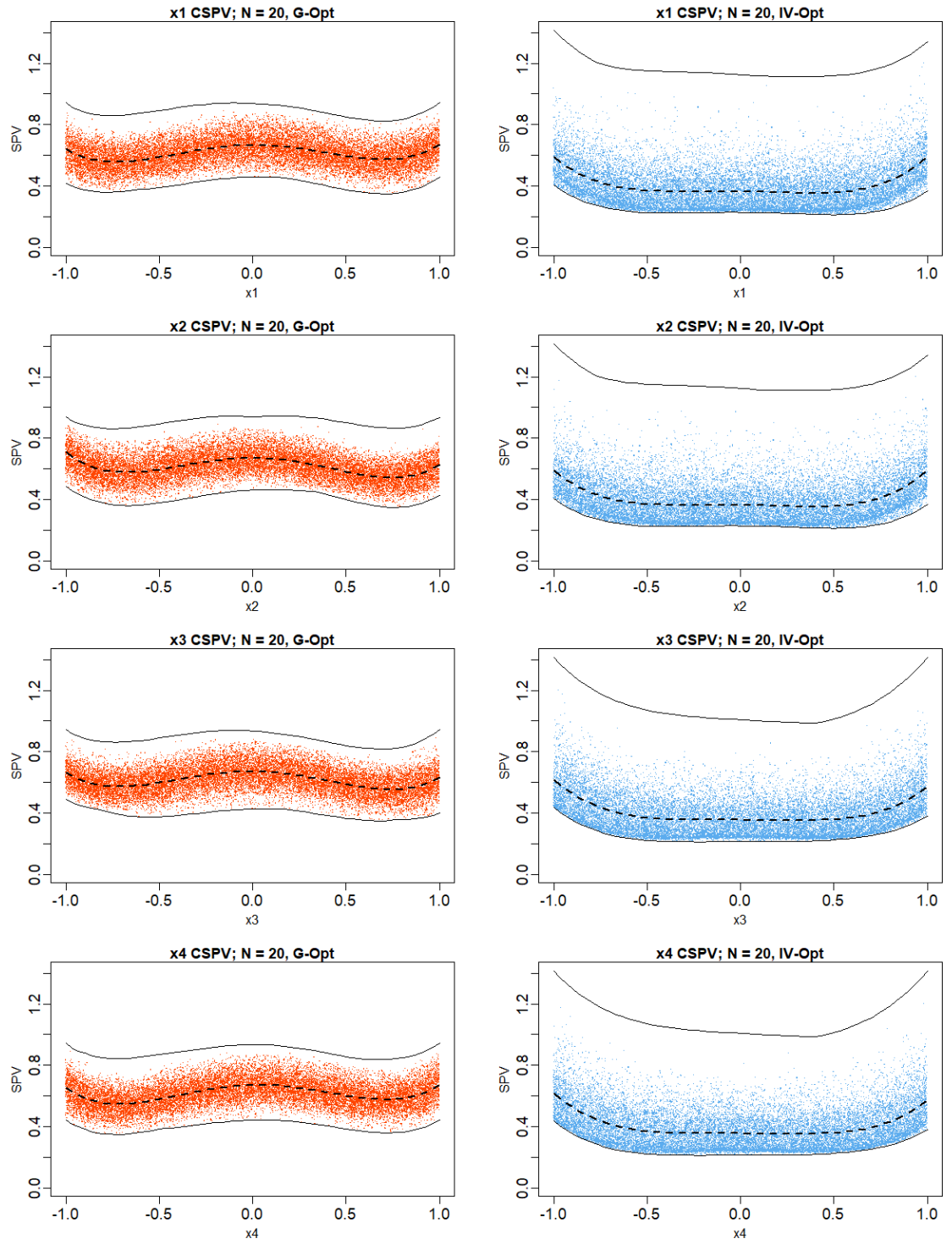


Figure 5.6: CSPV plots for $K = 4$, $N = 20$ comparing the G -PSO to IV -PSO design.

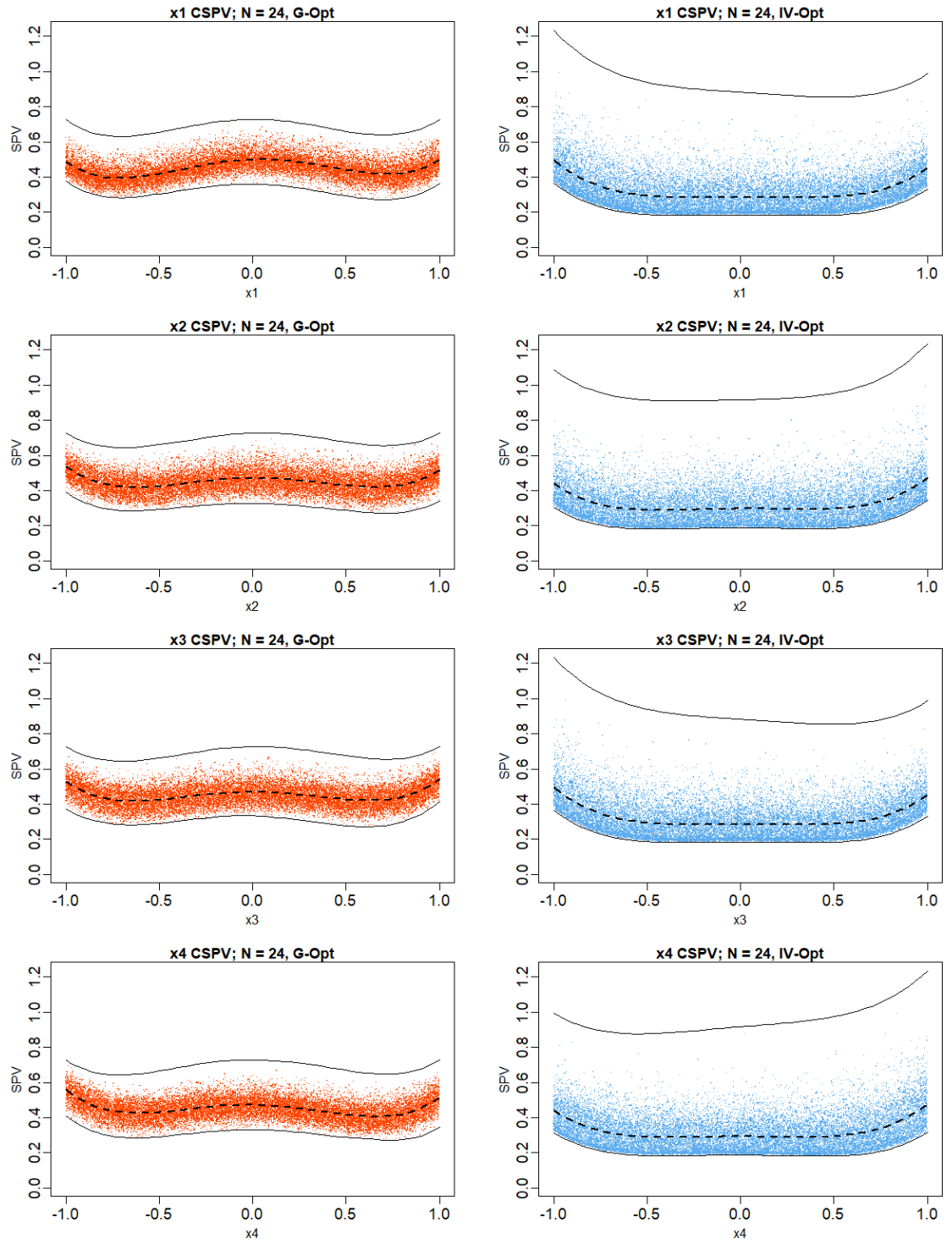


Figure 5.7: CSPV plots for $K = 4$, $N = 24$ comparing the G -PSO to IV -PSO design.

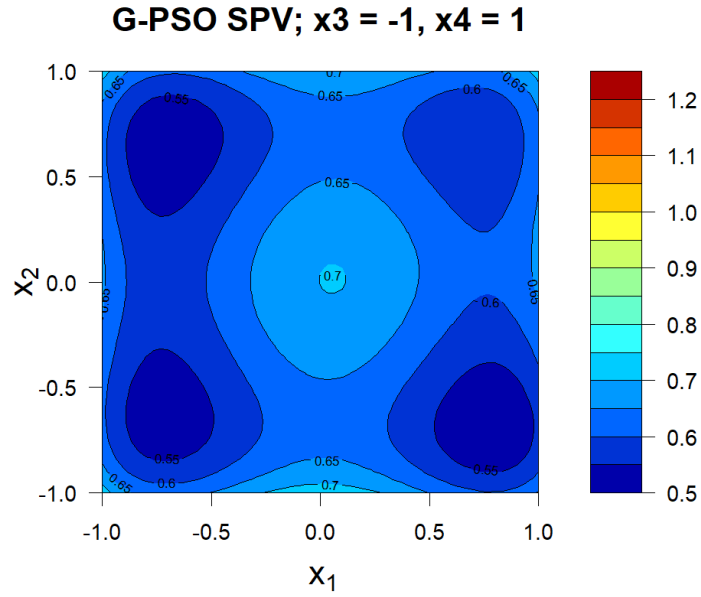
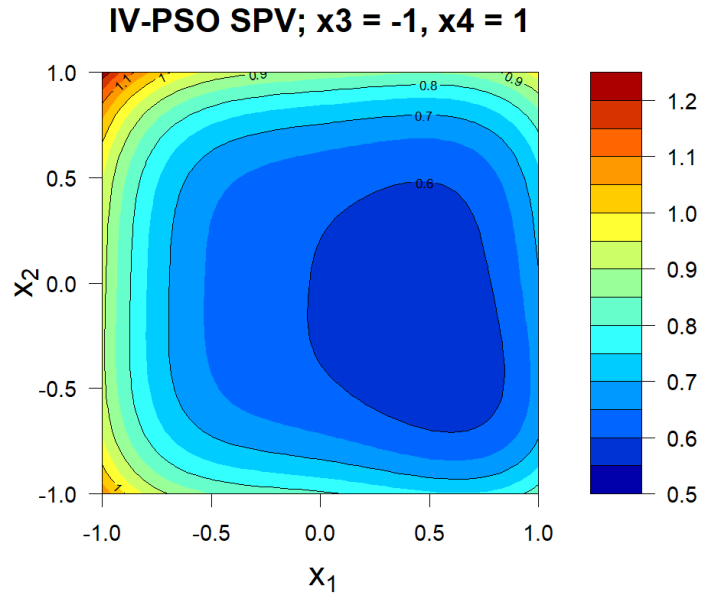
(a) *G*-PSO design SPV ($x_3 = -1, x_4 = 1$)(b) *IV*-PSO design SPV ($x_3 = -1, x_4 = 1$)

Figure 5.8: Contour plots of the face of $\mathcal{X}$ defined by $x_3 = -1, x_4 = 1$ for $N = 24$ (a) the four factor *G*-PSO design and (b) the four factor *IV*-PSO design.

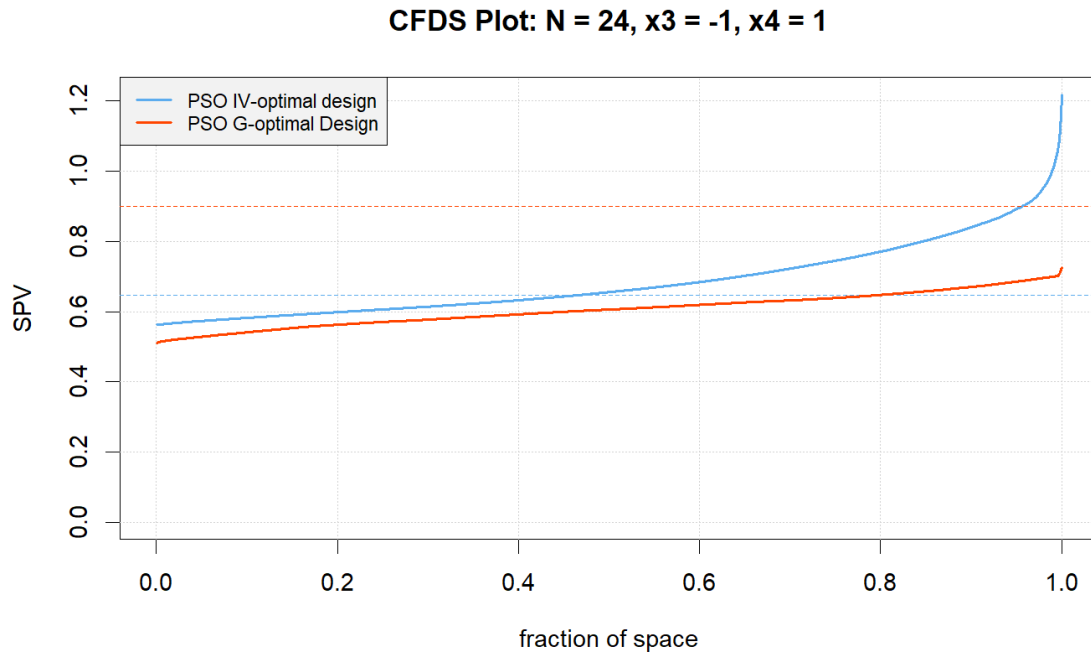


Figure 5.9: CFDS plot of the G and IV -PSO designs for face $(x_3 = -1, x_4 = 1)$. The dashed lines are the global IV -scores for both designs.

Table 5.7: Full space of degree-of-freedom scenarios on ν_1 (LOF) and ν_2 (PE) for a design problem on $K = 3$ factors with $N = 16$ runs, intending to support the second order model which has $p = 10$ parameters.

<i>df</i> scenario	replication scenario	M	$\nu_1 = M - 10$	$\nu_2 = 16 - M$
1	all points unique, no reps	16	6	0
2	one point replicated twice	15	5	1
3	one point replicated thrice, or two points each replicated twice	14	4	2
4	one point replicated four times, ..., or three points each replicated twice	13	3	3
5	one point replicated five times, ..., or four points each replicated twice	12	2	4
6	one point replicated six times, ..., or four points each replicated (three once each, one thrice)	11	1	5
7	one point replicated seven times, ..., or six points replicated twice	10	0	6

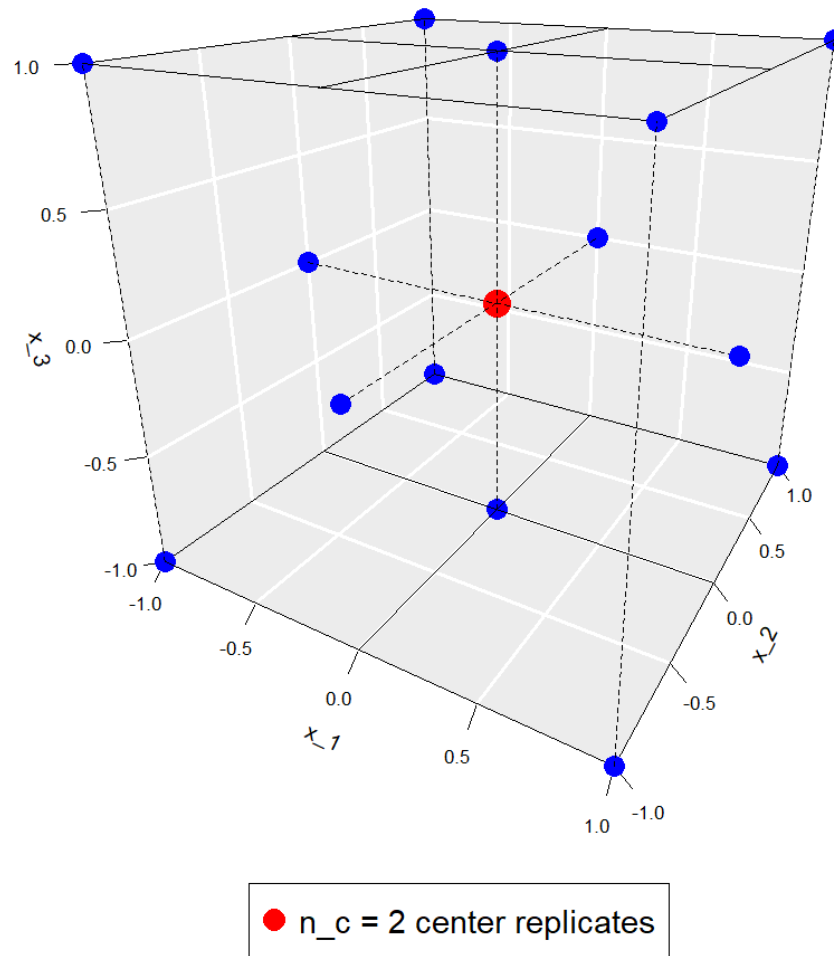
$K = 3, N = 16$ CCD

Figure 5.10: CCD with two center replicates for $K = 3$ factors and $N = 16$ design points.

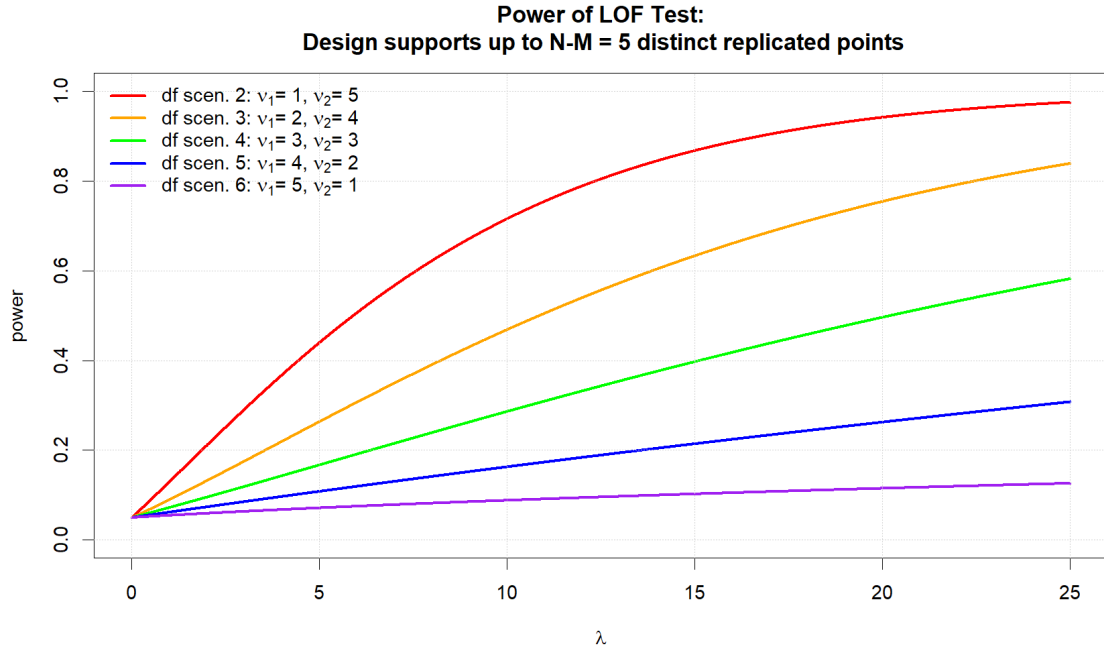


Figure 5.11: Power curves for the LOF test under the df scenarios shown in Table 5.7.

Table 5.8: The $K = 2, 3, 4$ and N scenarios for which we implemented exhaustive study of generating IV -optimal designs with specified replication using PSO.

K	N	# of replication scenarios (i.e. # of PSO searches)
2	10	12
	12	29
3	14	12
	16	29
4	20	18

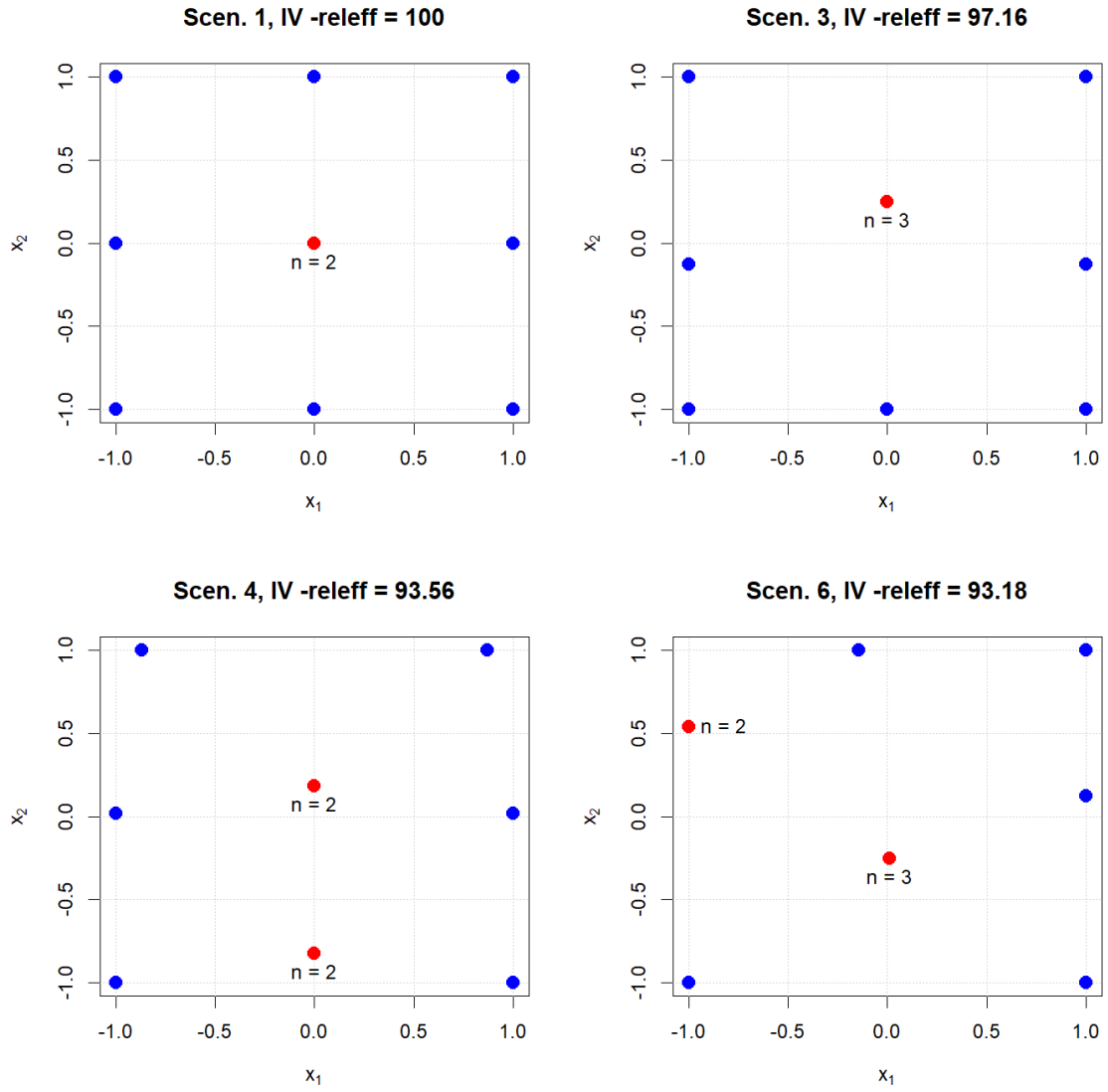


Figure 5.12: $K = 2$, $N = 10$ study: IV-PSO designs with specified replication corresponding to scenarios 1, 3, 4, and 6.

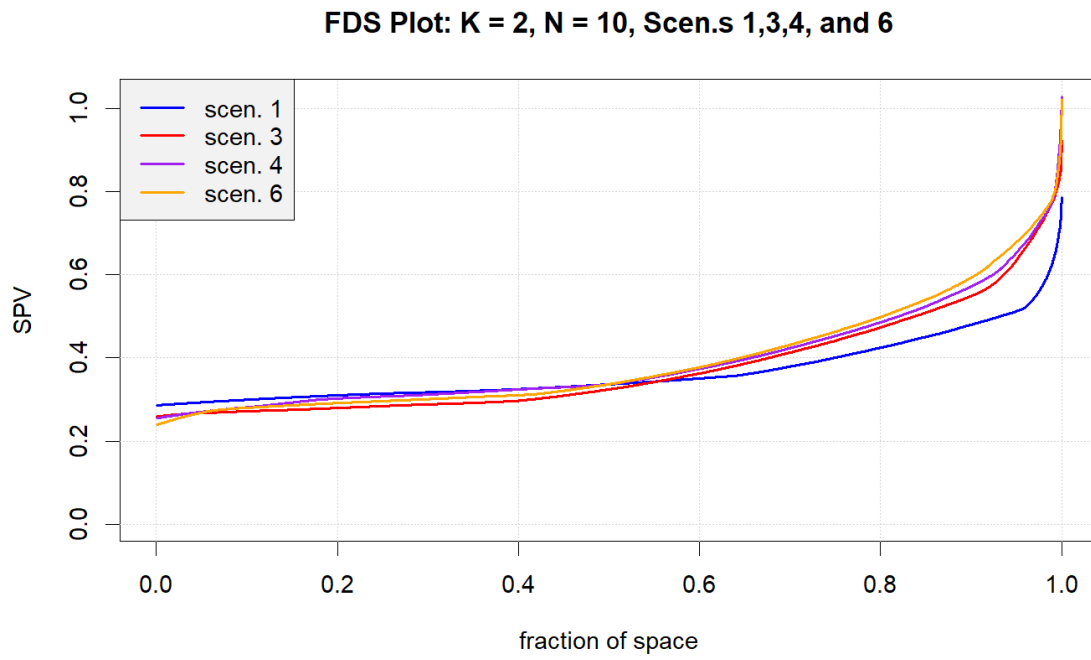
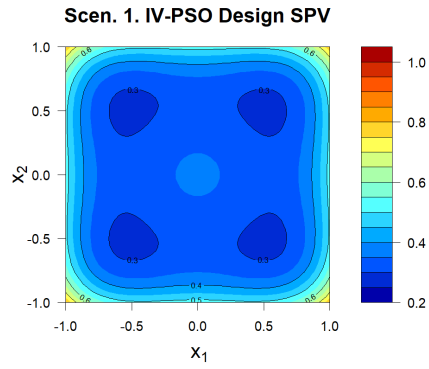
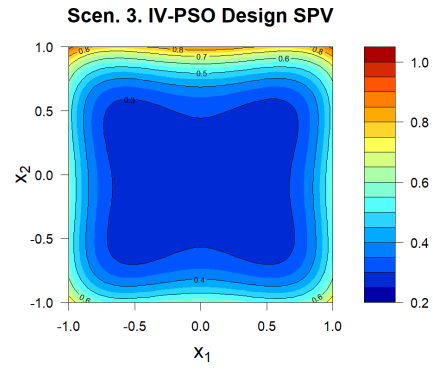


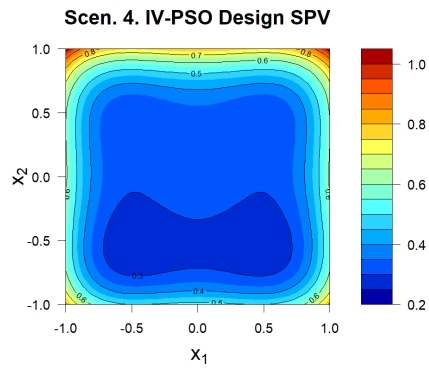
Figure 5.13: $K = 2$, $N = 10$ study: FDS plots of *IV*-PSO designs with specified replication corresponding to scenarios 1, 3, 4, and 6.



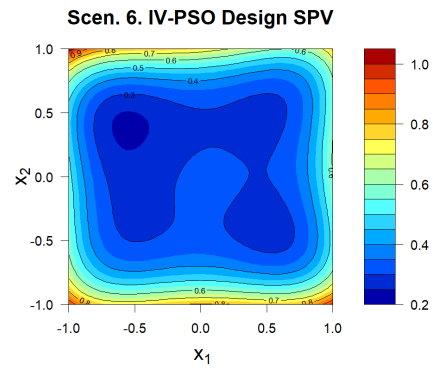
(a) Scenario 1.



(b) Scenario 3.



(c) Scenario 4.



(d) Scenario 6.

Figure 5.14: SPV surfaces for the $K = 2$ $N = 10$ selected IV-PSO design replication scenarios.

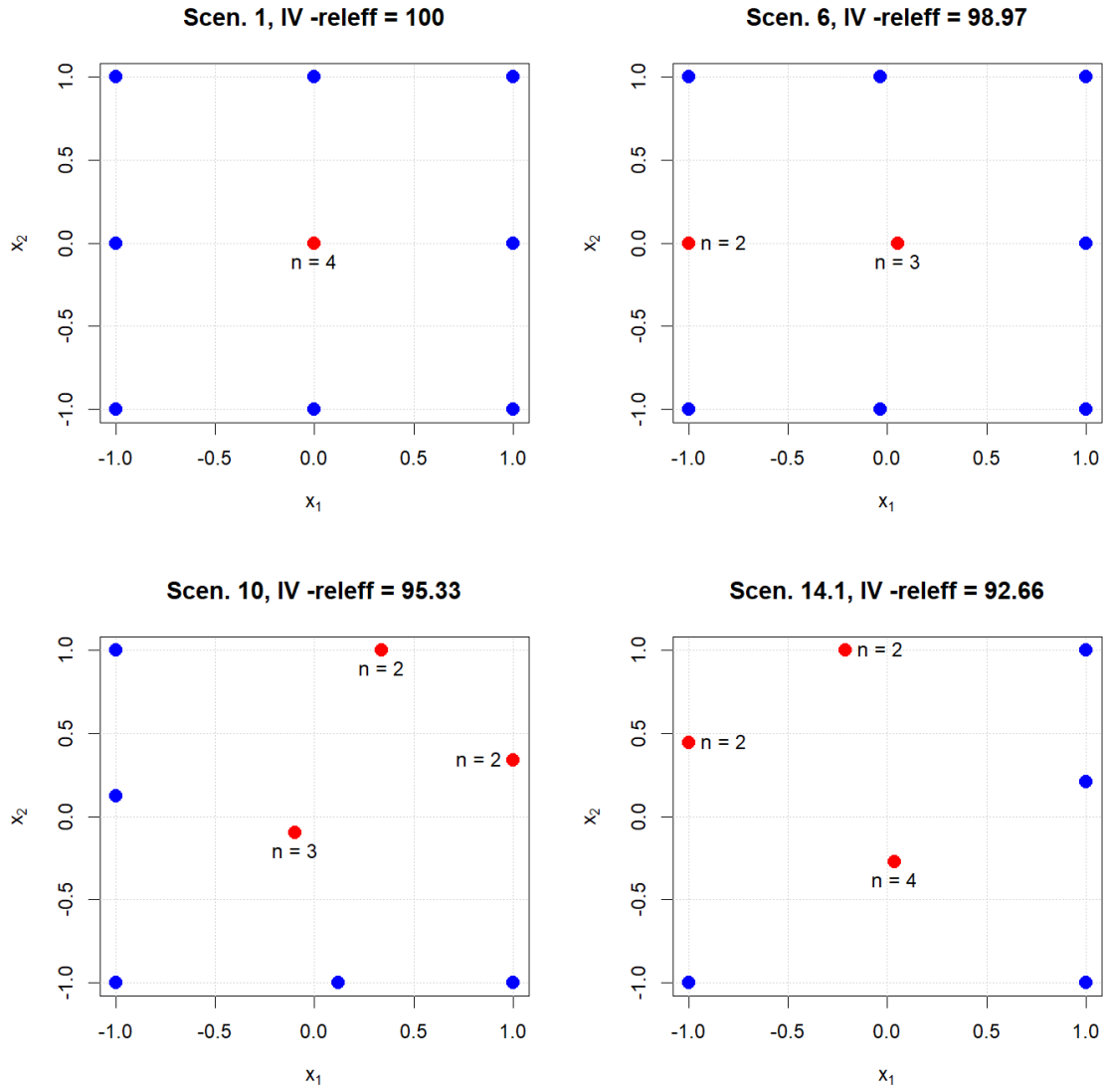


Figure 5.15: $K = 2$, $N = 12$ study: IV-PSO designs with specified replication corresponding to scenarios 1, 6, 10 and 14.1.

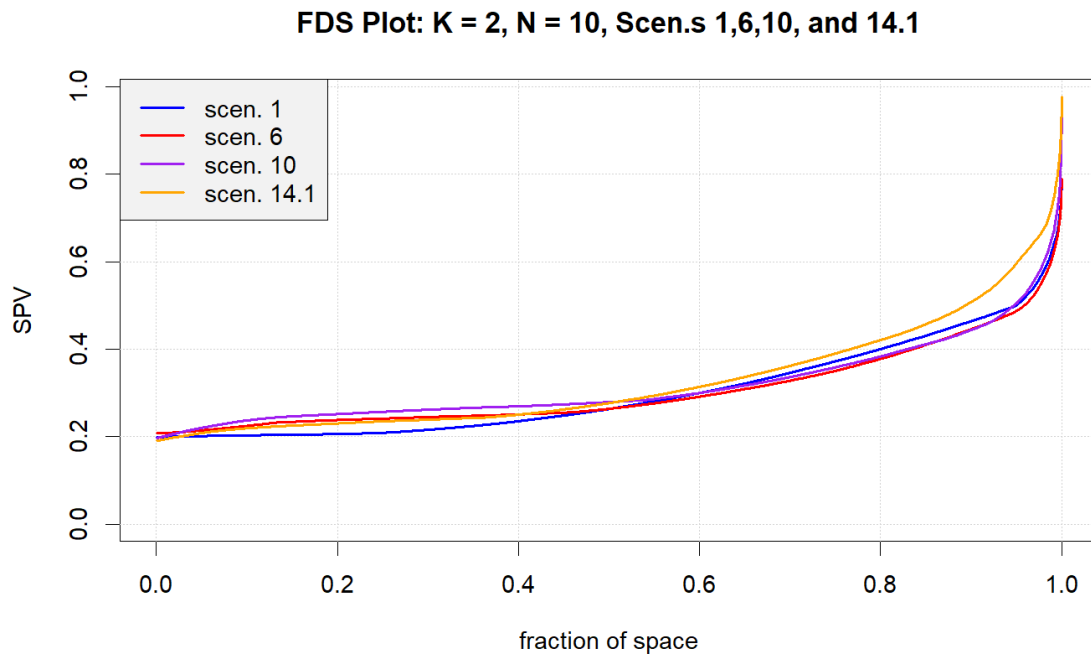
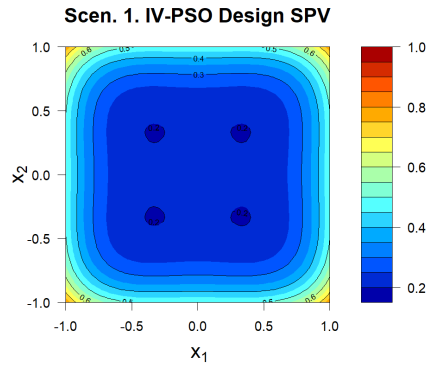
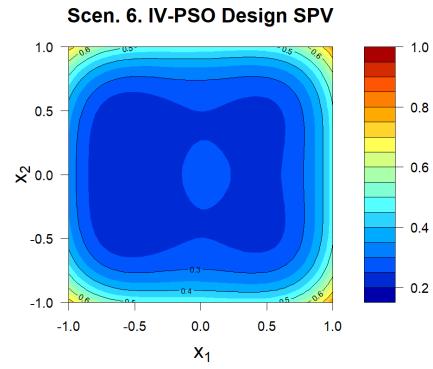


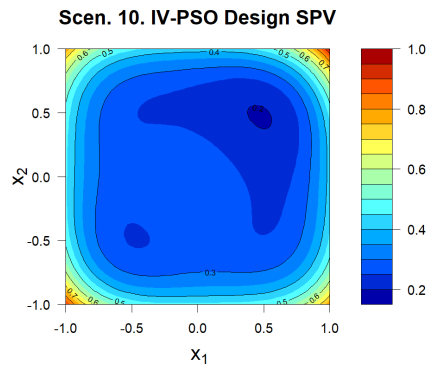
Figure 5.16: $K = 2$, $N = 12$ study: FDS plots of *IV*-PSO designs with specified replication corresponding to scenarios 1, 6, 10 and 14.1.



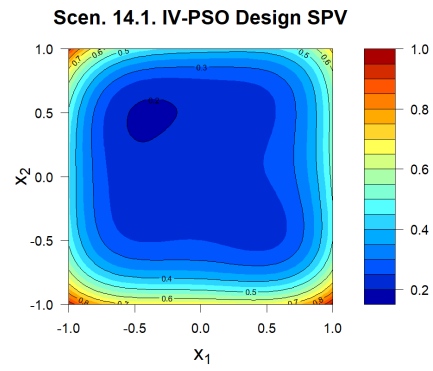
(a) Scenario 1.



(b) Scenario 6.



(c) Scenario 10.



(d) Scenario 14.1.

Figure 5.17: SPV surfaces for the $K = 2$ $N = 12$ selected IV-PSO design replication scenarios.

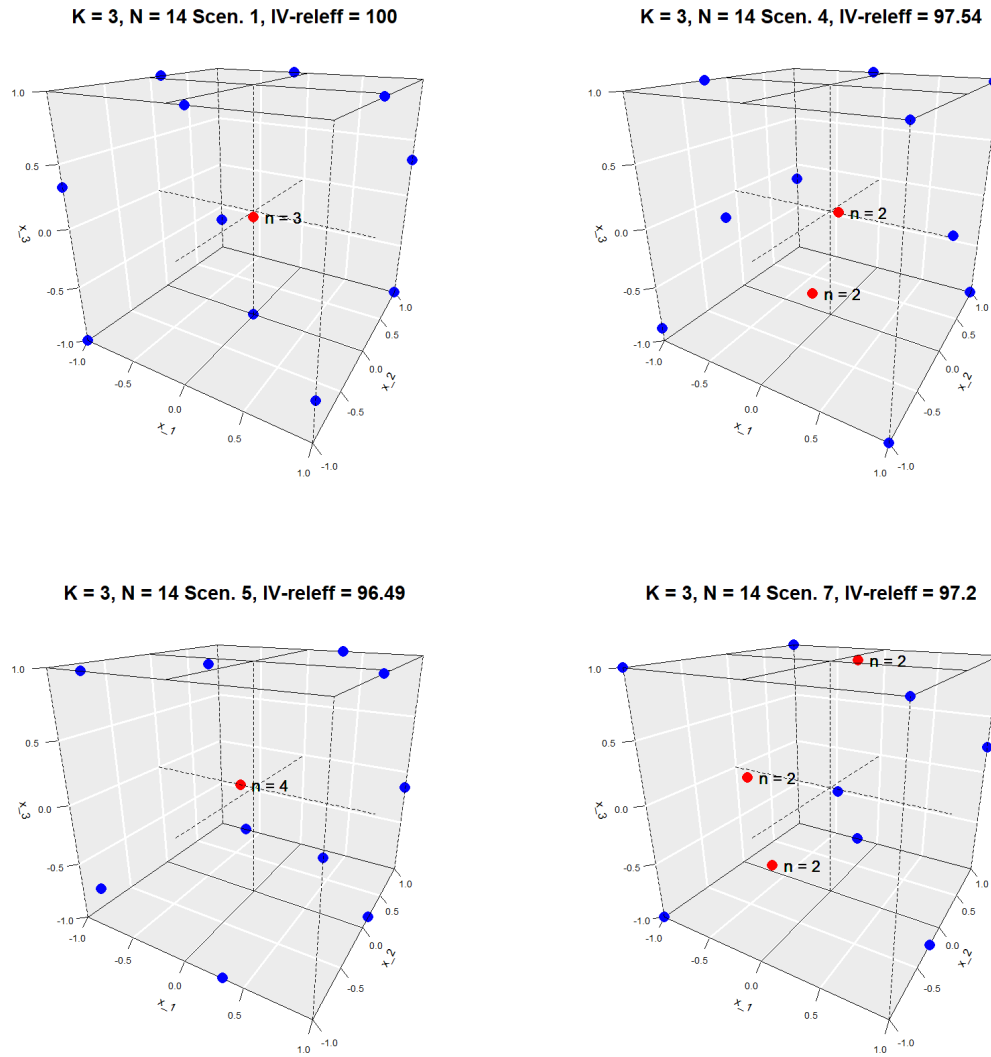


Figure 5.18: $K = 3$, $N = 14$ study: *IV*-PSO designs with specified replication corresponding to scenarios 1, 4, 5 and 7.

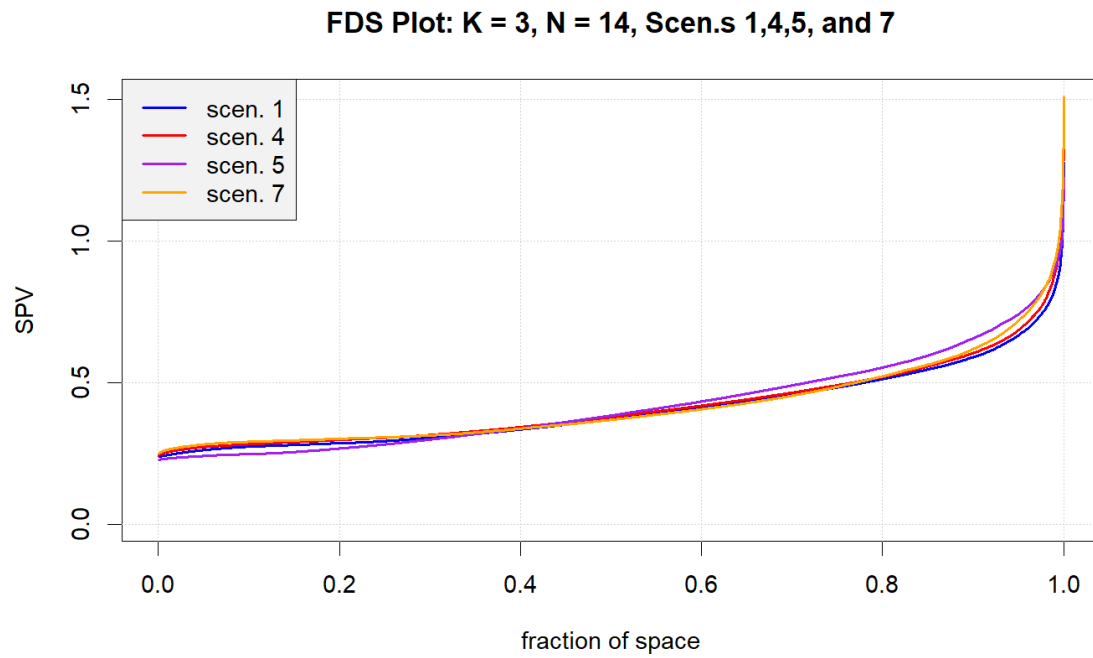


Figure 5.19: $K = 3$, $N = 14$ study: FDS plots of *IV*-PSO designs with specified replication corresponding to scenarios 1, 4, 5 and 7.

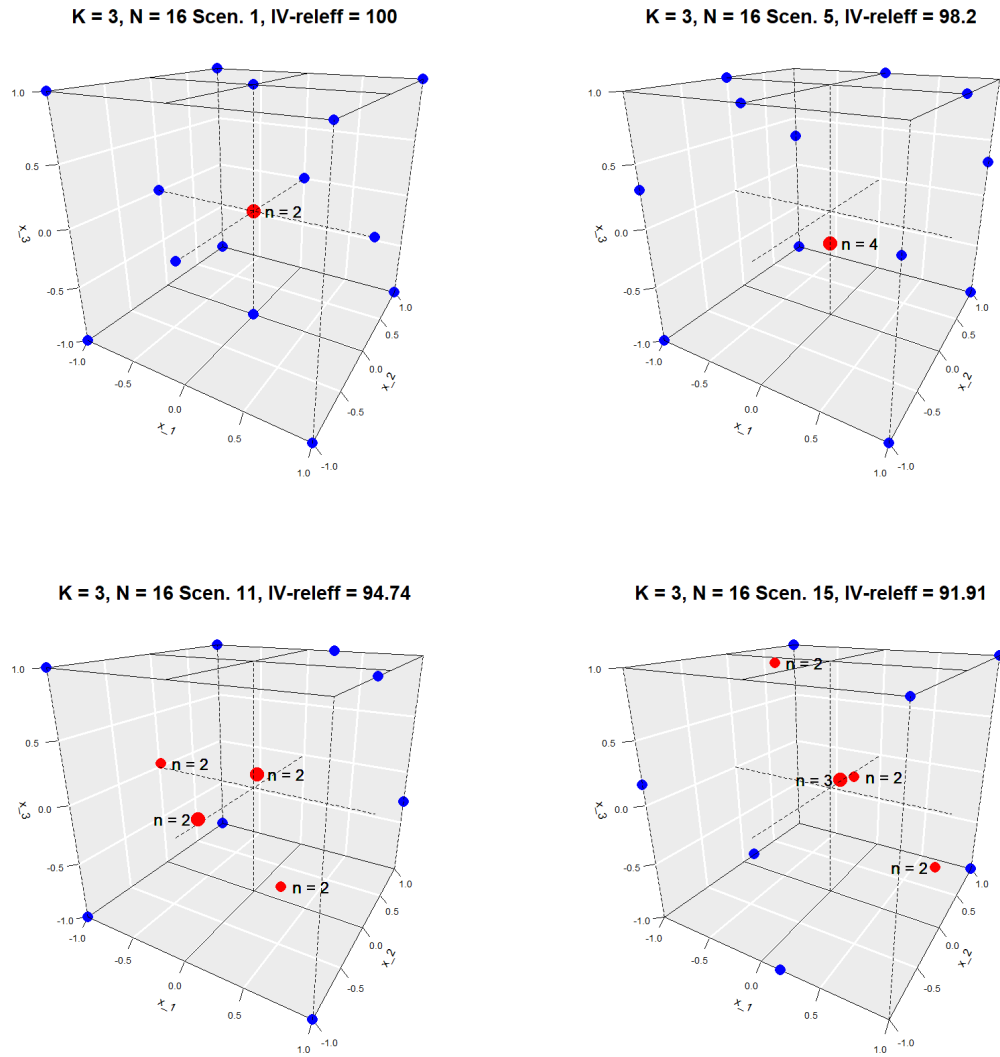


Figure 5.20: $K = 3$, $N = 16$ study: *IV*-PSO designs with specified replication corresponding to scenarios 1, 5, 11, and 15.

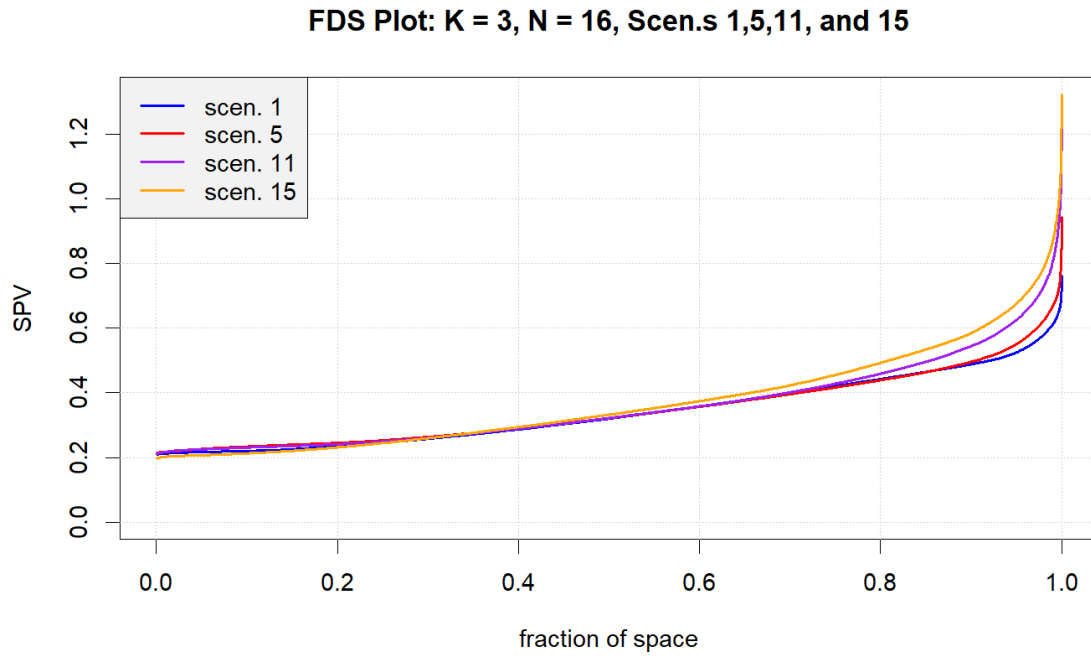


Figure 5.21: $K = 3$, $N = 16$ study: FDS plots of IV -PSO designs with specified replication corresponding to scenarios 1, 5, 11, and 15.

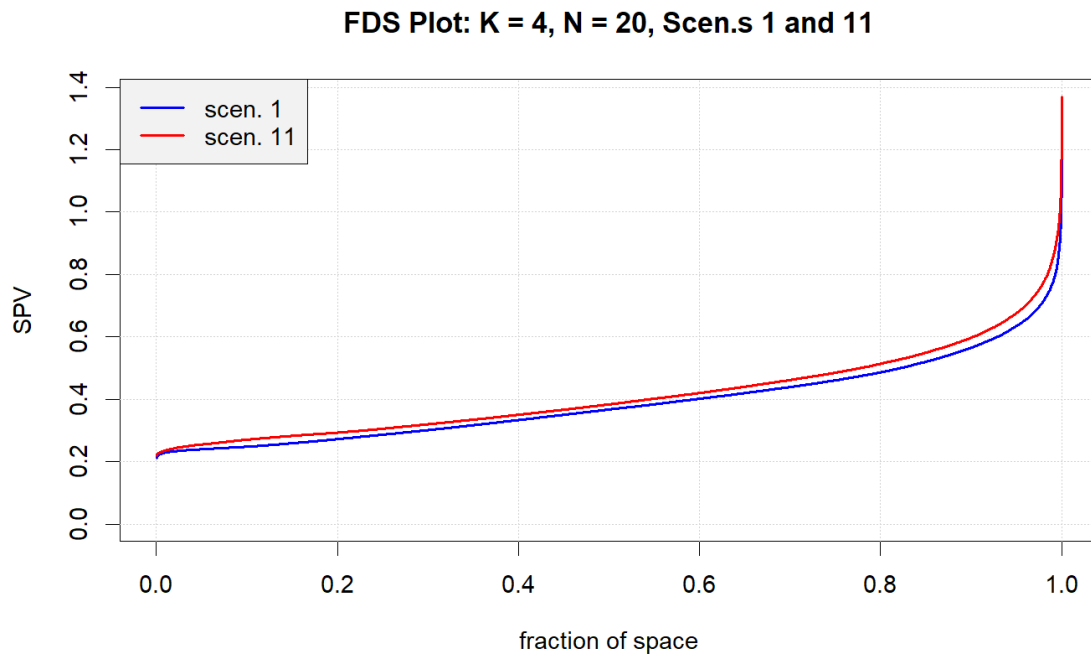


Figure 5.22: $K = 4$, $N = 20$ study: FDS plots of *IV*-PSO designs with specified replication corresponding to scenarios 1 and 11.

Table 5.9: The $K = 4$ $N = 20$ IV-PSO generated design, replication scenario 1.

$IV = 0.3894$				
n_{rep}	x_1	x_2	x_3	x_4
1	-1	0.08397	1	1
1	1	-1	0.57352	1
1	-0.22375	1	0.22663	1
1	1	0.43526	-1	1
1	-0.57423	-1	-1	1
1	-1	1	-1	0.40943
1	1	1	1	0.40568
1	0.09100	0.01158	1	-0.01161
1	-1	0.01317	-0.09117	-0.01242
1	-1	-1	1	-0.08430
1	1	-1	-1	-0.43545
1	1	1	-0.07635	-1
1	1	-0.40625	1	-1
1	-1	1	1	-1
1	-0.22355	-1	0.22655	-1
1	-1	-0.40963	-1	-1
1	0.07563	1	-1	-1
3	0.08485	-0.01162	-0.08596	0.01153

Table 5.10: The $K = 4$ $N = 20$ IV -PSO generated design, scenario 11.

$IV = 0.4133$				
n_{rep}	x_1	x_2	x_3	x_4
1	-1	0.44351	-1	1
1	1	1	0.00227	1
1	-1	-1	0.43811	1
1	-1	1	1	0.44359
1	1	0.05573	-1	-1
1	1	-1	1	0.05289
1	-1	-0.47615	1	-1
1	0.22759	-1	-0.00162	-1
1	0.75173	-1	-1	1
1	-1	-1	-1	-0.47509
1	0.74779	1	1	-1
1	-1	1	-0.43885	-1
2	0.12653	0.01701	1	1
2	-0.21438	-0.03648	-0.00053	-0.03612
2	0.71524	0.00328	0.00053	0.00223
2	0.12685	1	-1	0.01999

Table 5.11: Replication scenarios and *IV*-scores of designs generated by PSO for $K = 2, N = 10$.

K = 2, N = 10						Theoretical DF			IV-Crit	IV-releff
index	des.scenario	n.unique.max (M)	n.repeated	n.replicate (Ni)	rep.vector.set	SSE (N - p)	SSLOF (M - p)	SSPE (N - M)		
1	1	10	0	0	NULL	4	4	0	0.365873	100
2	2	9	1	2	x9, x9	4	3	1	0.365873	100
3	3	8	1	3	x8, x8, x8	4	2	2	0.376560	97.16
4	4		2	2	x7, x7, x8, x8				0.391052	93.56
5	5	7	1	4	x7, x7, x7, x7	4	1	3	0.402284	90.95
6	6		2	2, 3	x6, x6, x7, x7, x7				0.392646	93.18
7	7		3	2	x5, x5, x6, x6, x7, x7				0.409618	89.32
8	8	6	1	5	x6, x6, x6, x6, x6	4	0	4	0.454084	80.57
9	9.1		2	2, 4	x5, x5, x6, x6, x6, x6				0.425179	86.05
10	9.2			3, 3	x5, x5, x5, x6, x6, x6				0.442242	82.73
11	10		3	2, 2, 3	x4, x4, x5, x5, x6, x6, x6				0.409618	89.32
12	11		4	2, 2, 2, 2	x3, x3, x4, x4, x5, x5, x6, x6				0.431080	84.87

Table 5.12: Replication scenarios and IV -scores of designs generated by PSO for $K = 2, N = 12$.

K = 2, N = 12						Theoretical DF			IV-Crit	IV-releff
index	des.scenario	n.unique.max (M)	n.repeated	n.replicate (Ni)	rep.vector.set	SSE (N - p)	SSLOF (M - p)	SSPE (N - M)		
1	1	12	0	0	NULL	6	6	0	0.302778	100
2	2	11	1	2	x11, x11	6	5	1	0.302778	100
3	3	10	1	3	x10, x10, x10	6	4	2	0.302778	100
4	4		2	2	x9, x9, x10, x10				0.302778	100
5	5	9	1	4	x9, x9, x9, x9	6	3	3	0.302778	100
6	6		2	2, 3	x8, x8, x9, x9, x9				0.305917	98.97
7	7		3	2	x7, x7, x8, x8, x9, x9				0.315322	96.02
8	8	8	1	5	x8, x8, x8, x8, x8	6	2	4	0.328967	92.04
9	9.1		2	2, 4	x7, x7, x8, x8, x8, x8				0.320721	94.41
10	9.2			3, 3	x7, x7, x7, x8, x8, x8				0.334085	90.63
11	10		3	2, 2, 3	x6, x6, x7, x7, x8, x8, x8				0.317600	95.33
12	11		4	2	x5, x5, x6, x6, x7, x7, x8, x8				0.325342	93.06
13	12	7	1	6	x7, x7, x7, x7, x7, x7	6	1	5	0.370966	81.62
14	13.1		2	2, 5	x6, x6, x7, x7, x7, x7, x7				0.342293	88.46
15	13.2			3, 4	x6, x6, x6, x7, x7, x7, x7				0.346648	87.34
16	14.1		3	2, 2, 4	x5, x5, x6, x6, x7, x7, x7, x7				0.326760	92.66
17	14.2			2, 3, 3	x5, x5, x6, x6, x6, x7, x7, x7				0.342094	88.51
18	15		4	2, 2, 2, 3	x4, x4, x5, x5, x6, x6, x7, x7, x7				0.329344	91.93
19	16		5	2, 2, 2, 2, 2	x3, x3, x4, x4, x5, x5, x6, x6, x7, x7				0.333341	90.83
20	17	6	1	7	x6, x6, x6, x6, x6, x6, x6	6	0	6	0.430956	70.26
21	18.1		2	2, 6	x5, x5, x6, x6, x6, x6, x6				0.391403	77.36
22	18.2			3, 5	x5, x5, x5, x6, x6, x6, x6, x6				0.388482	77.94
23	18.3			4, 4	x5, x5, x5, x5, x6, x6, x6, x6				0.400519	75.6
24	19.1		3	2, 2, 5	x4, x4, x5, x5, x6, x6, x6, x6, x6				0.355836	85.09
25	19.2			2, 3, 4	x4, x4, x5, x5, x5, x6, x6, x6, x6				0.359653	84.19
26	20.1		4	2, 2, 2, 4	x3, x3, x4, x4, x5, x5, x6, x6, x6, x6				0.333341	90.83
27	20.2			2, 2, 3, 3	x3, x3, x4, x4, x5, x5, x5, x6, x6, x6				0.349434	86.65
28	21		5	2, 2, 2, 2, 3	x2, x2, x3, x3, x4, x4, x5, x5, x6, x6, x6				0.340807	88.84
29	22		6	2	x1, x1, x2, x2, x3, x3, x4, x4, x5, x5, x6, x6				0.382827	79.09

Table 5.13: Replication scenarios and *IV*-scores of designs generated by PSO for $K = 3, N = 14$.

K = 3 , N = 14						Theoretical DF			IV-Crit	IV-releff
index	des.scenario	n.unique.max (M)	n.repeated	n.replicate (Ni)	rep.vector.set	SSE (N - p)	SSLOF (M - p)	SSPE (N - M)		
1	1	14	0	0	NULL	4	4	0	0.406495	100
2	2	13	1	2	x13, x13	4	3	1	0.406495	100
3	3	12	1	3	x12, x12, x12	4	2	2	0.406495	100
4	4		2	2	x11, x11, x12, x12				0.416761	97.54
5	5	11	1	4	x11, x11, x11, x11	4	1	3	0.421271	96.49
6	6		2	2, 3	x10, x10, x11, x11, x11				0.424353	95.79
7	7		3	2	x9, x9, x10, x10, x11, x11				0.418226	97.2
8	8	10	1	5	x10, x10, x10, x10, x10	4	0	4	0.463915	87.62
9	9.1		2	2, 4	x9, x9, x10, x10, x10, x10				0.439339	92.52
10	9.2			3, 3	x9, x9, x9, x10, x10, x10				0.454061	89.52
11	10		3	2, 2, 3	x8, x8, x9, x9, x10, x10, x10				0.433128	93.85
12	11		4	2	x7, x7, x8, x8, x9, x9, x10, x10				0.440478	92.28

Table 5.14: Replication scenarios and *IV*-scores of designs generated by PSO for $K = 3, N = 16$.

K = 3, N = 16						Theoretical DF			IV-Crit	IV-releff
index	des.scenario	n.unique.max (M)	n.repeated	n.replicate (Ni)	rep.vector.set	SSE (N - p)	SSLOF (M - p)	SSPE (N - M)		
1	1	16	0	0	NULL	6	6	0	0.340517	100
2	2	15	1	2	x15, x15	6	5	1	0.340517	100
3	3	14	1	3	x14, x14, x14	6	4	2	0.346762	98.2
4	4		2	2	x13, x13, x14, x14				0.346762	98.2
5	5	13	1	4	x13, x13, x13, x13	6	3	3	0.346762	98.2
6	6		2	2, 3	x12, x12, x13, x13, x13				0.356284	95.57
7	7		3	2	x11, x11, x12, x12, x13, x13				0.355140	95.88
8	8	12	1	5	x12, x12, x12, x12, x12	6	2	4	0.364791	93.35
9	9.1		2	2, 4	x11, x11, x12, x12, x12, x12				0.363584	93.66
10	9.2			3, 3	x11, x11, x11, x12, x12, x12				0.375106	90.78
11	10		3	2, 2, 3	x10, x10, x11, x11, x12, x12, x12				0.361812	94.11
12	11		4	2	x9, x9, x10, x10, x11, x11, x12, x12				0.359414	94.74
13	12	11	1	6	x11, x11, x11, x11, x11, x11	6	1	5	0.391317	87.02
14	13.1		2	2, 5	x10, x10, x11, x11, x11, x11				0.377274	90.26
15	13.2			3, 4	x10, x10, x10, x11, x11, x11, x11				0.382267	89.08
16	14.1		3	2, 2, 4	x9, x9, x10, x10, x11, x11, x11, x11				0.372611	91.39
17	14.2			2, 3, 3	x9, x9, x10, x10, x10, x11, x11, x11				0.377650	90.17
18	15		4	2, 2, 2, 3	x8, x8, x9, x9, x10, x10, x11, x11, x11				0.370506	91.91
19	16	10	1	7	x10, x10, x10, x10, x10, x10, x10	6	0	6	0.443613	76.76
20	17.1		2	2, 6	x9, x9, x10, x10, x10, x10, x10				0.409784	83.1
21	17.2			3, 5	x9, x9, x9, x10, x10, x10, x10, x10				0.407253	83.61
22	17.3			4, 4	x9, x9, x9, x9, x10, x10, x10, x10				0.417651	81.53
23	18.1		3	2, 2, 5	x8, x8, x9, x9, x10, x10, x10, x10				0.397005	85.77
24	18.2			2, 3, 4	x8, x8, x9, x9, x9, x10, x10, x10, x10				0.394343	86.35
25	18.3			3, 3, 3	x8, x8, x8, x9, x9, x9, x10, x10, x10				0.389726	87.37
26	19.1		4	2, 2, 2, 4	x7, x7, x8, x8, x9, x9, x10, x10, x10, x10				0.389696	87.38
27	19.2			2, 2, 3, 3	x7, x7, x8, x8, x9, x9, x9, x10, x10, x10				0.386655	88.07
28	20		5	2, 2, 2, 2, 3	x6, x6, x7, x7, x8, x8, x9, x9, x10, x10, x10				0.384921	88.46
29	21		6	2	x5, x5, x6, x6, x7, x7, x8, x8, x9, x9, x10, x10				0.392546	86.75

Table 5.15: Replication scenarios and *IV*-scores of designs generated by PSO for $K = 4$, $N = 20$.

K = 4, N = 20						Theoretical DF				IV-Crit	IV-releff
index	des.scenario	n.unique.max (M)	n.repeated	n.replicate (Ni)	rep.vector.set	SSE (N - p)	SSLOF (M - p)	SSPE (N - M)			
1	1	20	0	0	NULL	5	5	0		0.389388	100
2	2	19	1	2	x19, x19	5	4	1		0.389388	100
3	3	18	1	3	x18, x18, x18	5	3	2		0.389388	100
4	4		2	2	x17, x17, x18, x18					0.390131	99.81
5	5	17	1	4	x17, x17, x17, x17	5	2	3		0.390130	99.81
6	6		2	2, 3	x16, x16, x17, x17, x17					0.399774	97.4
7	7		3	2	x15, x15, x16, x16, x17, x17					0.401048	97.09
8	8	16	1	5	x16, x16, x16, x16, x16	5	1	4		0.401022	97.1
9	9		2	3, 3	x15, x15, x15, x16, x16, x16					0.413123	94.25
10	10		3	2, 2, 3	x14, x14, x15, x15, x16, x16, x16					0.402900	96.65
11	11		4	2	x13, x13, x14, x14, x15, x15, x16, x16					0.413251	94.23
12	12	15	1	6	x15, x15, x15, x15, x15, x15	5	0	5		0.430786	90.39
13	13.1		2	2, 5	x14, x14, x15, x15, x15, x15					0.432661	90
14	13.2			3, 4	x14, x14, x14, x15, x15, x15, x15					0.421985	92.28
15	14.1		3	2, 2, 4	x13, x13, x14, x14, x15, x15, x15, x15					0.419295	92.87
16	14.2			2, 3, 3	x13, x13, x14, x14, x14, x15, x15, x15					0.441634	88.17
17	15		4	2, 2, 2, 3	x12, x12, x13, x13, x14, x14, x15, x15, x15					0.428436	90.89
18	16		5	2	x11, x11, x12, x12, x13, x13, x14, x14, x15, x15					0.435196	89.47

Appendix

In this Appendix we construct the matrix form of the decomposition of the sum-of-square error (*SSE*) component of a linear model into the pure-error (*PE*) and lack-of-fit (*LOF*) sum-of-squares. Further, as no references we are aware of show the distribution of the *LOF* statistic under the alternative hypothesis, we derive this distribution and produce power curves for the *LOF* test.

The modeling is as follows. Let $\mathbf{y} : N \times 1$ be the vector of response measurements. We assume $\mathbf{y} \sim \mathcal{N}_N(\boldsymbol{\mu}, \sigma^2 \mathbf{I}_N)$. The researcher will assume a linear model as a function of covariates $\mathbf{X} : N \times K$. The model matrix is denoted $\mathbf{F} : N \times p$ has rank p with linear parameter vector $\boldsymbol{\beta} : p \times 1$. This leads to competing hypotheses:

$$H_0 : E(\mathbf{y}) = \mathbf{F}\boldsymbol{\beta} \quad (5.27)$$

vs.

$$H_A : E(\mathbf{y}) = \boldsymbol{\mu} \quad (5.28)$$

where the former expectation is the linear mean structure assumed by the analyst and the latter is some possibly true general mean structure. Suppose that in the assumed structure $\mathbf{F}$ there are M unique design points with N_i replicates per group. Thus $\sum_{i=1}^M N_i = N$. If the replicated design points are grouped by row in $\mathbf{F}$ then it is possible to write the model matrix as

$$\mathbf{F} = \begin{pmatrix} \mathbf{1}_{N_1} \mathbf{f}'_1 \\ \mathbf{1}_{N_2} \mathbf{f}'_2 \\ \vdots \\ \mathbf{1}_{N_M} \mathbf{f}'_M \end{pmatrix} \quad (5.29)$$

where $\mathbf{f}'_i : 1 \times N_i$.

The scalar forms of the SS terms are given in [68] as:

$$SSE = \sum_{i=1}^M \sum_{j=1}^{N_i} (y_{ij} - \hat{y}_i)^2 \quad (5.30)$$

$$SSPE = \sum_{i=1}^M \sum_{j=1}^{N_i} (y_{ij} - \bar{y}_i)^2 \quad (5.31)$$

$$SSLOF = \sum_{i=1}^M N_i (\bar{y}_i - \hat{y}_i)^2. \quad (5.32)$$

And it is claimed that

$$SSE = SSLOF + SSPE. \quad (5.33)$$

A test statistic for lack-of-fit of model $\mathbf{F}$ is given in [68] as

$$F_0 = \frac{SSLOF/(M-p)}{SSPE/(N-M)} = \frac{MSLOF}{MSPE} \quad (5.34)$$

and under H_0 it is claimed that $F_0 \sim F_{M-p, N-M}$. The distribution of F_0 under H_A is not given.

The objective of this section is

1. Find vector-matrix forms for SSE , $SSLOF$, and $SSPE$.
2. Construct the distribution of F_0 under $H_A : E(\mathbf{y}) = \boldsymbol{\mu}$.

In order to identify matrix forms of the SS terms and prove the error decomposition in Eq. 5.33 the following definitions, theorems, and propositions are required. First, we require the properties of idempotent matrices as shown in reference [10]:

Definition 1. Let $\mathbf{P} : n \times n$ be a square matrix. If $\mathbf{P}^2 = \mathbf{P}$ then $\mathbf{P}$ is said to be idempotent.

Theorem 2. (Idempotent Matrices). Let $\mathbf{P}$ be an $n \times n$ matrix.

(a) If $\mathbf{P}^2 = \mathbf{P}$ then $\text{rank}(\mathbf{P}) = \text{trace}(\mathbf{P})$.

(b) $\mathbf{P}^2 = \mathbf{P} \iff (\mathbf{I}_n - \mathbf{P})^2 = \mathbf{I}_n - \mathbf{P}$

Let $\mathbf{J}_{N_i} : N_i \times N_i$ represent a matrix of 1's. Let $\oplus$ represent the direct sum operator.

Proposition 1. *The symmetric matrix*

$$\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$$

is idempotent.

Proof. First we will show that $\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$ is idempotent.

$$\left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right)^2 = \bigoplus_{i=1}^M N_i^{-2} \mathbf{J}_{N_i} \mathbf{J}_{N_i}.$$

Where $\mathbf{J}_{N_i} \mathbf{J}_{N_i} = \{N_i\} : N_i \times N_i$. Thus

$$\left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right)^2 = \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}.$$

This demonstrates that $\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$ idempotent by definition. It follows from Theorem 2 (b) that

$$\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$$

is idempotent. ■

Proposition 2.

$$\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F} = \mathbf{F}$$

Proof.

$$\begin{aligned}
 \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F} &= \begin{pmatrix} N_1^{-1} \mathbf{J}_{N_1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & N_2^{-1} \mathbf{J}_{N_2} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & N_M^{-1} \mathbf{J}_{N_M} \end{pmatrix} \begin{pmatrix} \mathbf{1}_{N_1} \mathbf{f}'_1 \\ \mathbf{1}_{N_2} \mathbf{f}'_2 \\ \vdots \\ \mathbf{1}_M \mathbf{f}'_M \end{pmatrix} \\
 &= \begin{pmatrix} N_1^{-1} \mathbf{J}_{N_1} \mathbf{1}_{N_1} \mathbf{f}'_1 \\ N_2^{-1} \mathbf{J}_{N_2} \mathbf{1}_{N_2} \mathbf{f}'_2 \\ \vdots \\ N_M^{-1} \mathbf{J}_{N_M} \mathbf{1}_M \mathbf{f}'_M \end{pmatrix}
 \end{aligned}$$

Noting that

$$\mathbf{J}_{N_i} \mathbf{1}_{N_i} = N_i \mathbf{1}_{N_i}$$

it follows that

$$\begin{aligned}
 \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F} &= \begin{pmatrix} N_1^{-1} \mathbf{J}_{N_1} \mathbf{1}_{N_1} \mathbf{f}'_1 \\ N_2^{-1} \mathbf{J}_{N_2} \mathbf{1}_{N_2} \mathbf{f}'_2 \\ \vdots \\ N_M^{-1} \mathbf{J}_{N_M} \mathbf{1}_M \mathbf{f}'_M \end{pmatrix} \\
 &= \begin{pmatrix} \mathbf{1}_{N_1} \mathbf{f}'_1 \\ \mathbf{1}_{N_2} \mathbf{f}'_2 \\ \vdots \\ \mathbf{1}_M \mathbf{f}'_M \end{pmatrix} \\
 &= \mathbf{F}
 \end{aligned}$$

■

The hat matrix related to model $\mathbf{F}\boldsymbol{\beta}$ is defined as the $N \times N$ matrix

$$\mathbf{H} = \mathbf{F} (\mathbf{F}'\mathbf{F})^{-1} \mathbf{F}'. \quad (5.35)$$

$\mathbf{H}$ is a perpendicular projection operator and so it is both idempotent and symmetric [10]. Further, the fitted/predicted values are

$$\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}. \quad (5.36)$$

Last, the rank of $\mathbf{H}$ is the same as the rank of $\mathbf{F}$, namely p .

Proposition 3.

$$\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{H} = \mathbf{H}$$

Proof.

$$\begin{aligned} \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{H} &= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F} (\mathbf{F}'\mathbf{F})^{-1} \mathbf{F}' \\ &= \mathbf{F} (\mathbf{F}'\mathbf{F})^{-1} \mathbf{F}' && \text{(applied Proposition 2)} \\ &= \mathbf{H} && \text{(definition of } \mathbf{H} \text{)} \end{aligned}$$

■

Proposition 4. *The symmetric matrix*

$$\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H}$$

is idempotent.

Proof.

$$\begin{aligned}
\left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right)^2 &= \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \\
&= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} + \mathbf{H} \mathbf{H} - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{H} - \mathbf{H} \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \\
&= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} + \mathbf{H} - 2 \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{H} && \text{(idempotency and sym.)} \\
&= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} + \mathbf{H} - 2\mathbf{H} && \text{(Proposition 3)} \\
&= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H}
\end{aligned}$$

■

Proposition 5. *The product of*

$$\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$$

and

$$\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H}$$

is $\mathbf{0}$.

Proof.

$$\begin{aligned}
&\left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \\
&= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} + \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{H} \\
&= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} + \mathbf{H} && \text{(idempotency and prop. 3)} \\
&= \mathbf{0}
\end{aligned}$$

■

Proposition 6.

$$\text{rank} \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) = N - M$$

Proof. Noting that Proposition 1 shows that the matrix is idempotent, we can apply Theorem 2 part (b).

$$\begin{aligned} \text{rank} \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) &= \text{trace} \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \\ &= \text{trace}(\mathbf{I}_N) - \text{trace} \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \\ &= N - \sum_{i=1}^M N_i^{-1} \text{trace}(\mathbf{J}_{N_i}) \\ &= N - \sum_{i=1}^M N_i^{-1} N_i \\ &= N - \sum_{i=1}^M 1 \\ &= N - M \end{aligned}$$

■

Proposition 7.

$$\text{rank} \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) = M - p$$

Proof. Noting that Proposition 4 shows that the matrix is idempotent, we can apply

Theorem 2 part (b).

$$\begin{aligned}
 \text{rank} \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) &= \text{trace} \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \\
 &= \text{trace} \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) - p \\
 &= \sum_{i=1}^M N_i^{-1} \text{trace}(\mathbf{J}_{N_i}) - p \\
 &= \sum_{i=1}^M N_i^{-1} N_i - p \\
 &= \sum_{i=1}^M 1 - p \\
 &= M - p
 \end{aligned}$$

■

Proposition 8.

$$SSLOF = \mathbf{y}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y}$$

Proof. We will show that the proposed form reduces to the scalar form of Eq. 5.32.

Appealing to the idempotency of

$$\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H}$$

we can write

$$\mathbf{y}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} = \left(\left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} \right)' \left(\left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} \right).$$

Now,

$$\begin{aligned}
 \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} &= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{y} - \mathbf{H} \mathbf{y} \\
 &= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{y} - \hat{\mathbf{y}} \\
 &= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_M \end{pmatrix} - \begin{pmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \\ \hat{\mathbf{y}}_M \end{pmatrix}.
 \end{aligned}$$

Noting that

$$N_i^{-1} \mathbf{J}_{N_i} \mathbf{y}_i = N_i^{-1} \mathbf{1}_{N_i} \sum_{j=1}^{N_i} y_{ij} = \bar{y}_i \mathbf{1}_{N_i}$$

and

$$\hat{\mathbf{y}}_i = \hat{y}_i \mathbf{1}_{N_i}$$

as all points in a replication group are subject to the same model matrix row $\mathbf{f}_i'$ and, hence, get the same predicted value, we can write

$$\left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} = \begin{pmatrix} (\bar{y}_1 - \hat{y}_1) \mathbf{1}_{N_1} \\ (\bar{y}_2 - \hat{y}_2) \mathbf{1}_{N_2} \\ \vdots \\ (\bar{y}_M - \hat{y}_M) \mathbf{1}_{N_M} \end{pmatrix}.$$

Thus,

$$\begin{aligned}
& \mathbf{y}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} \\
&= \begin{pmatrix} (\bar{y}_1 - \hat{y}_1) \mathbf{1}_{N_1} & (\bar{y}_2 - \hat{y}_2) \mathbf{1}_{N_2} & \cdots & (\bar{y}_M - \hat{y}_M) \mathbf{1}_{N_M} \end{pmatrix}' \begin{pmatrix} (\bar{y}_1 - \hat{y}_1) \mathbf{1}_{N_1} \\ (\bar{y}_2 - \hat{y}_2) \mathbf{1}_{N_2} \\ \vdots \\ (\bar{y}_M - \hat{y}_M) \mathbf{1}_{N_M} \end{pmatrix} \\
&= \sum_{i=1}^M (\bar{y}_i - \hat{y}_i)^2 \mathbf{1}_{N_i}' \mathbf{1}_{N_i} \\
&= \sum_{i=1}^M N_i (\bar{y}_i - \hat{y}_i)^2
\end{aligned}$$

■

Proposition 9.

$$SSPE = \mathbf{y}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y}$$

Proof. We will show that the proposed form reduced to the scalar form of Eq. 5.31.

Appealing to the idempotency of

$$\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$$

we can write

$$\mathbf{y}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} = \left(\left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} \right)' \left(\left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} \right).$$

Now,

$$\left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} = \begin{pmatrix} \mathbf{y}_1 - \mathbf{1}_{N_1} \bar{y}_1 \\ \mathbf{y}_2 - \mathbf{1}_{N_2} \bar{y}_2 \\ \vdots \\ \mathbf{y}_M - \mathbf{1}_{N_M} \bar{y}_M \end{pmatrix}.$$

Thus,

$$\begin{aligned}
& \mathbf{y}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} \\
&= \left(\mathbf{y}_1 - \mathbf{1}_{N_1} \bar{y}_1 \quad \mathbf{y}_2 - \mathbf{1}_{N_2} \bar{y}_2 \quad \cdots \quad \mathbf{y}_M - \mathbf{1}_{N_M} \bar{y}_M \right)' \begin{pmatrix} \mathbf{y}_1 - \mathbf{1}_{N_1} \bar{y}_1 \\ \mathbf{y}_2 - \mathbf{1}_{N_2} \bar{y}_2 \\ \vdots \\ \mathbf{y}_M - \mathbf{1}_{N_M} \bar{y}_M \end{pmatrix} \\
&= \sum_{i=1}^M (\mathbf{y}_i - \mathbf{1}_{N_i} \bar{y}_i)' (\mathbf{y}_i - \mathbf{1}_{N_i} \bar{y}_i)
\end{aligned}$$

where

$$\begin{aligned}
& (\mathbf{y}_i - \mathbf{1}_{N_i} \bar{y}_i)' (\mathbf{y}_i - \mathbf{1}_{N_i} \bar{y}_i) \\
&= \mathbf{y}_i' \mathbf{y}_i - \mathbf{y}_i' \mathbf{1}_{N_i} \bar{y}_i - \mathbf{1}_{N_i}' \bar{y}_i \mathbf{y}_i + \mathbf{1}_{N_i}' \mathbf{1}_{N_i} \bar{y}_i^2 \\
&= \sum_{j=1}^{N_i} y_{ij}^2 - 2N_i \bar{y}_i^2 + N_i \bar{y}_i^2 \\
&= \sum_{j=1}^{N_i} y_{ij}^2 - N_i \bar{y}_i^2 \\
&= \sum_{j=1}^{N_i} (y_{ij} - \bar{y}_i)^2
\end{aligned}$$

where the last line above is proved in Theorem 5.2.4 pp 212 of [22]. Thus, we have

$$\mathbf{y}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} = \sum_{i=1}^M \sum_{j=1}^{N_i} (y_{ij} - \bar{y}_i)^2$$

which is the required result. ■

Proposition 10.

$$SSE = SSLOF + SSPE$$

Proof. First note that it is a well known result that

$$SSE = \mathbf{y}' (\mathbf{I}_N - \mathbf{H}) \mathbf{y}.$$

Now,

$$\begin{aligned} SSLOF + SSPE &= \mathbf{y}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{y} + \mathbf{y}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} \\ &= \mathbf{y}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} + \mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{y} \\ &= \mathbf{y}' (\mathbf{I}_N - \mathbf{H}) \mathbf{y} \end{aligned}$$

■

Next, we will construct the distribution of the F -statistic shown in Eq. 5.34.

We require the following theorem from reference [67]:

Theorem 3. (Distribution of Quadratic Forms). *Let $\mathbf{y} \sim \mathcal{N}_N(\boldsymbol{\mu}, \sigma^2 \mathbf{I}_N)$. Let $N \times N$ square matrices $\mathbf{Q}_1$ and $\mathbf{Q}_2$ both be symmetric, positive semi-definite, and idempotent. If $\mathbf{Q}_1 \mathbf{Q}_2 = \mathbf{0}$ then:*

(a)

$$\frac{\mathbf{y}' \mathbf{Q}_1 \mathbf{y}}{\sigma^2} \sim \chi^2_{\left(\nu_1 = \text{rank}(\mathbf{Q}_1), \lambda_1 = \frac{\boldsymbol{\mu}' \mathbf{Q}_1 \boldsymbol{\mu}}{\sigma^2} \right)}$$

(b)

$$\frac{\mathbf{y}' \mathbf{Q}_2 \mathbf{y}}{\sigma^2} \sim \chi^2_{\left(\nu_2 = \text{rank}(\mathbf{Q}_2), \lambda_2 = \frac{\boldsymbol{\mu}' \mathbf{Q}_2 \boldsymbol{\mu}}{\sigma^2} \right)}$$

(c) $\mathbf{y}' \mathbf{Q}_1 \mathbf{y}$ and $\mathbf{y}' \mathbf{Q}_2 \mathbf{y}$ are independent statistics.

Further, if

$$\boldsymbol{\mu}' \mathbf{Q}_2 \boldsymbol{\mu} = 0$$

then

(d)

$$\frac{\mathbf{y}' \mathbf{Q}_1 \mathbf{y} / \text{rank}(\mathbf{Q}_1)}{\mathbf{y}' \mathbf{Q}_2 \mathbf{y} / \text{rank}(\mathbf{Q}_2)} \sim F_{\left(\nu_1 = \text{rank}(\mathbf{Q}_1), \nu_2 = \text{rank}(\mathbf{Q}_2), \lambda_1 = \frac{\boldsymbol{\mu}' \mathbf{Q}_1 \boldsymbol{\mu}}{\sigma^2}\right)}$$

Let $\mathbf{Q}_1 = \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H}$ and $\mathbf{Q}_2 = \mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i}$. We have $\mathbf{Q}_1 \mathbf{Q}_2 = \mathbf{0}$ by Proposition 5. Therefore, the corresponding quadratic forms of *SSLOF* and *SSPE* are chi-squared distributed as in Theorem 3.

Proposition 11. *Under both H_0 and H_A*

$$\boldsymbol{\mu}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \boldsymbol{\mu} = 0$$

Proof. Let H_0 be true, then $E(\mathbf{y}) = \boldsymbol{\mu} = \mathbf{F}\boldsymbol{\beta}$. It follows that

$$\begin{aligned} \boldsymbol{\mu}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \boldsymbol{\mu} &= \boldsymbol{\beta}' \mathbf{F}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \mathbf{F} \boldsymbol{\beta} \\ &= \boldsymbol{\beta}' \left(\mathbf{F}' \mathbf{F} - \mathbf{F}' \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F} \right) \boldsymbol{\beta} \\ &= \boldsymbol{\beta}' (\mathbf{F}' \mathbf{F} - \mathbf{F}' \mathbf{F}) \boldsymbol{\beta} \\ &= \boldsymbol{\beta}' \mathbf{0} \boldsymbol{\beta} \\ &= 0 \end{aligned}$$

where the third line above is an application of Proposition 2.

Now let H_A be true, then $E(\mathbf{y}) = \boldsymbol{\mu}$. First, note we can write $\boldsymbol{\mu}$ as repeated

means in the M groups as

$$\boldsymbol{\mu} = \begin{pmatrix} \mathbf{1}_{N_1}\mu_1 \\ \mathbf{1}_{N_2}\mu_2 \\ \vdots \\ \mathbf{1}_{N_M}\mu_M \end{pmatrix}.$$

Then we note that

$$\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \boldsymbol{\mu} = \boldsymbol{\mu}$$

by an argument similar to Proposition 2. Now,

$$\begin{aligned} \boldsymbol{\mu}' \left(\mathbf{I}_N - \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \right) \boldsymbol{\mu} &= \boldsymbol{\mu}' \boldsymbol{\mu} - \boldsymbol{\mu}' \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \boldsymbol{\mu} \\ &= \boldsymbol{\mu}' \boldsymbol{\mu} - \boldsymbol{\mu}' \boldsymbol{\mu} \\ &= 0 \end{aligned}$$

■

Proposition 12. *The statistic*

$$F_0 = \frac{SSLOF/(M-p)}{SSPE/(N-M)}$$

follows a non-central F distribution parametrized as

$$F \left(\nu_1=M-p, \nu_2=N-M, \lambda_1 = \frac{\boldsymbol{\mu}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \boldsymbol{\mu}}{\sigma^2} \right)$$

Proof. Let $\mathbf{Q}_1$ and $\mathbf{Q}_2$ be as defined above and apply Theorem 3 and Propositions 6 and 7. ■

Proposition 13. *When H_0 in 5.21 is true the statistic*

$$F_0 = \frac{SSLOF/(M-p)}{SSPE/(N-M)}$$

follows a central F distribution parametrized as

$$F_{(\nu_1=M-p, \nu_2=N-M)}$$

Proof. Let H_0 be true then $E(\mathbf{y}) = \boldsymbol{\mu} = \mathbf{F}\boldsymbol{\beta}$. Consider the result of Proposition 12 and consider the non-centrality parameter

$$\begin{aligned} \lambda_1 &= \frac{\boldsymbol{\mu}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \boldsymbol{\mu}}{\sigma^2} \\ &= \frac{\boldsymbol{\beta}' \mathbf{F}' \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{F} \boldsymbol{\beta}}{\sigma^2} \end{aligned}$$

and noting that by Proposition 2 we have

$$\begin{aligned} \left(\bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} - \mathbf{H} \right) \mathbf{F} &= \bigoplus_{i=1}^M N_i^{-1} \mathbf{J}_{N_i} \mathbf{F} - \mathbf{H} \mathbf{F} \\ &= \mathbf{F} - \mathbf{F} \\ &= \mathbf{0} \end{aligned}$$

the desired result follows. ■

CHAPTER SIX

A NEW PARTICLE SWARM OPTIMIZATION ALGORITHM BASED ON THE
 AITCHISON GEOMETRY FOR GENERATING EXACT OPTIMAL MIXTURE
 EXPERIMENT DESIGNS

ABSTRACT

In mixture experiments the design variables are no longer on an absolute or relative scale (e.g.time, temperature, pressure), rather, each of the K components represents a proportion of the total. This constraint creates a number of technical differences in the geometry of the design space, the linear modeling, and algorithms used to generate optimal designs. In this chapter, we develop an approach to adapting particle swarm optimization for optimizing functions with matrix inputs defined on the Cartesian product of standard K -simplices. To accomplish this we take advantage of the Aitchison geometry, a non-Euclidean geometry defined on the standard K -simplex. We demonstrate that, in this geometry, specific definitions of vector addition and scalar multiplication induce a vector space on the K -simplex. Then we adapt these definitions to the update equations in PSO. We demonstrate via application to numerous known D and IV -optimal designs that the resulting PSO algorithm can efficiently generate these designs for various models. Last, we extend the approach to design generation when the mixture components are subject to upper and lower bounds.

6.1 Introduction

Experiments on mixtures are common in the food and materials sciences where the objective is to identify the best recipe, i.e.the best mixture of components, to

achieve an optimal response. For example, Cornell (2011) [31] lists some simple examples of a mixture experiment:

1. (food) Produce a fruit punch that has a highly acceptable flavor, color and consumer acceptance from a blend of watermelon, pineapple, and orange juice.
2. (materials) Produce a highly efficacious pesticide via blending/mixing several chemicals.
3. (materials) Road flares are the product of blended proportions of magnesium, sodium nitrate, strontium nitrate, and a binder material. Identify that combination of components that maximizes the brightness and durability of the road flare.

As the components of mixture experiments are proportions, they sum to one. Thus the design space for K factors is the standard $(K - 1)$ -simplex which is defined as

$$\Delta^{K-1} := \{\mathbf{x}' \in \mathbb{R}^K \mid \mathbf{x}'\mathbf{1}_K = 1, 0 \leq \mathbf{x}'\mathbf{e}_i \leq 1 \text{ for } i = 1, \dots, K\} \quad (6.1)$$

where $\mathbf{e}_i$ is a vector of 0s with a 1 in the i th position. When designing an experiment using N points $\mathbf{x}' \in \Delta^{K-1}$ the designer is tasked with choosing some $N \times K$ matrix from

$$\mathbf{X} \in \bigtimes_{j=1}^N \Delta^{K-1}. \quad (6.2)$$

To add one more level of complication, in some design scenarios in mixture experiments, the factors are also constrained to lower and upper bounds on the proportions stated as

$$0 \leq L_i \leq \mathbf{x}'\mathbf{e}_i \leq U_i \leq 1 \quad \text{for } i = 1, 2, \dots, K. \quad (6.3)$$

Thus, in this setting, a general way of expressing the optimal design problem (for minimization objectives f) is to find that design $\mathbf{X}^*$ that

$$\begin{aligned}
 & \min_{\mathbf{X} \in \times_{j=1}^N \Delta^{K-1}} f(\mathbf{X}) \\
 \text{s.t. } & L_i \leq \mathbf{x}'_j \mathbf{e}_i \leq U_i \quad \forall i, j = 1, 2, \dots, N \\
 \text{and } & \mathbf{x}'_j \mathbf{1}_K = 1 \quad j = 1, 2, \dots, N.
 \end{aligned} \tag{6.4}$$

In this chapter we develop a PSO based algorithm to generate optimal designs under the formulation of expression 6.4.

In the optimization literature it is common to formulate the general optimization problem, for functions of vector inputs of dimension n , with non-linear inequality and equality constraints as

$$\begin{aligned}
 & \min_{\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n} f(\mathbf{x}) \\
 \text{s.t. } & g_i(\mathbf{x}) \leq 0 \text{ for } i = 1, 2, \dots, p \\
 \text{and } & h_j(\mathbf{x}) = 0 \text{ for } j = p + 1, \dots, m.
 \end{aligned} \tag{6.5}$$

A quick Google search for terms similar to “constrained particle swarm optimization” elucidates that a plethora of research literature entries address the problem of expression 6.5 from a particle swarm perspective. We will focus on a small collection of such literature that is close to the main vein of PSO algorithm development (some of which are authored by the progenitors of PSO) [21, 49, 50, 52, 64, 72, 90]. Among these publications we identified two distinct approaches to adapting PSO to the constrained optimization problem of expression 6.5:

1. transform the constrained optimization problem into an unconstrained search

by appealing to a penalized objective function which introduces a number of additional tuning parameters, or

2. introduce no new parameters but alter the PSO algorithm to check the constraints at each candidate point, at each iteration and only update personal, local and global best solutions if they live in the feasible region defined by the constraints.

The former approach is described in [64, 72, 90] in detail. The approach resembles a Lagrange multiplier approach by redefining the optimization objective to

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \lambda(k)H(g_1(\mathbf{x}), \dots, g_p(\mathbf{x}), h_{p+1}(\mathbf{x}), \dots, h_m(\mathbf{x})) \quad (6.6)$$

where ϕ is the unconstrained optimization objective, H is a penalty added to the objective if any of the g or h inequality and equality constraints (resp.) are violated by the candidate point, and λ is the penalty weight that changes with iteration k of the PSO algorithm. The appeal of this approach is that existing PSO codes can be (near seamlessly) used as the search space is not constrained. The addition of the penalty weight λ , however, requires some minor tweaks to the code as it needs to be allowed to increase during the PSO search. The penalty term generally increases as PSO iterates and so infeasible solutions are increasingly penalized as the search progresses. The rate of the penalty increase should follow the rate of the decrease of the swarm radius, though, so as to allow the swarm sufficient time to find the feasible region. It is apparently easier to implement the penalized objective approach for inequality constraints of the form g than for equality constraints h . Masuda, Kurihara, and Aiyoshi (2010) [64] illustrate this approach for only inequality constraints g and illustrate reasonable performance on a number of engineering optimization problems. They highlight that the only requirement is that λ increases monotonically during

the search and they give a simple expression for how to set this parameter. They do note, however, that the optimal value of the penalty parameter (e.g. in terms of convergence time) is problem dependent and may require some tweaking for each specific problem. The other papers in this set address the more complicated problem which includes the equality constraints h . The issue they all address is the fact that PSO initializes particles and updates particle velocity under random draws from continuous probability distributions. Because it is highly unlikely (in fact impossible) that initial particles and moving particles fall on the space defined by $h_j(\mathbf{x}) = 0$, they all recommend relaxing the equality constraint to create a set which does not have measure 0 by introducing a precision parameter ϵ and redefining the constraints as

$$|h_j(\mathbf{x})| - \epsilon \leq 0.$$

Parsopoulos and Vrahatis (2002) [72] illustrate the general approach to some very simple polynomial benchmark problems. They give little direction on how to select optimal values for the additional parameters and only illustrate simple functions for the penalty parameter: $\lambda(k) = \sqrt{k}$ and $\lambda(k) = k\sqrt{k}$. Takahama and Sakai (2005) [90] illustrate another application of the penalize approach to a more complicated benchmark problem which includes numerous equality constraints. They study in detail the slack parameter ϵ and their work indicates that the performance of the search algorithm, both in speed and effectiveness of finding the optimum, may be sensitive to this parameter. The reader may note that we did initial investigation of the approach shown in Masuda, Kurihara, and Aiyoshi (2010) [64] (which only deals with inequality constraints) and found a varying level of success to a number of published engineering optimization problems.

The authors of papers [21, 49, 50, 52] take the second approach (or a hybrid

of the two) to dealing with constraints in PSO by appealing not to alterations of the objective, but to the PSO algorithm itself in order to address the constrained optimization problem. These approaches either don't update the personal, local, or global bests if the candidate point is not in the feasible region [49, 50], or, they produce a relative ranking of importance, e.g. a point closer to the feasible region is a better candidate than one that is farther away [21, 52] and they use the constraints to build a ranking mechanism. Hu and Eberhart (2002) and Hu, Eberhart and Shi (2003) provide especially useful contributions (one reason is that these are the PSO progenitors). In these papers they address the constrained optimization problem by two adaptations to PSO:

- (particle initialization) If the initialized particle position is not in the feasible region, keep randomly initializing it until it is.
- (best position updates) Personal, local, and global best values in PSO are only counted and used in the algorithm if they exist in the feasible region.

These authors highlight that the appeal of this approach is its simplicity for how the constraints g and h are used in the algorithm and that there are no additional parameters introduced to the algorithm. They do warn, however, that it may take a long time to initialize particles in the feasible region if it is small, and, for the equality constraints h it is almost impossible to find a set of initial solutions in the feasible space. Further, as the particles move, if the feasible space is very small, it could be difficult for the searching particles to land in the feasible space and so it may take a long time to run the PSO algorithm before a solution is found.

The similarity between the general optimization problem shown in expression 6.5 and the optimal design problem for mixture experiments shown in expression 6.4 is immediately apparent. An additional technical detail that one would have to address

to adapting the current PSO literature to exact optimal mixture DoE is the fact that we have functions with matrix inputs (but this could easily be addressed in either approach). We chose, however, not to adapt this PSO literature to generating optimal designs for mixture experiments. The reason is due to multiple authors expressing:

1. the addition of the parameters required for the penalized objective approach can be problem dependent and may require fine-tuning for each problem, and
2. the equality constraints can be troublesome to deal with as this is a subset of $\mathbb{R}^K$ is a lot ‘less dense’ and therefore it is difficult to simulate from and move to via swarm initialization and the PSO velocity and position updates.

Based on these considerations, we develop a new PSO algorithm that incorporates the equality constraint $\mathbf{x}'\mathbf{1}_K = 1$ very naturally and rigorously into the definition of the search space and elegantly ensures that no particles will leave this subset of $\mathbb{R}^K$ as they progress searching for the optimum. We are intending to produce an algorithm that can be applied to any optimal mixture DoE problem and with a measurable degree of (stochastic) precision be expected to perform well without the additional need to fine tune new problem specific parameters.

6.2 Adapting the Aitchison geometry to enable PSO on the Cartesian product of standard $(K-1)$ -simplices.

6.2.1 Motivation

Recall from Chapter 3 that we solved the problem of adapting PSO to optimize functions with matrix inputs by defining the velocity and position update equations

as

$$\mathbf{V}_i(t+1) = \omega \mathbf{V}_i(t) \quad (\text{inertia}) \quad (6.7)$$

$$+ c_1 \mathbf{u} \odot (\mathbf{P}_{\text{best},i} - \mathbf{X}_i(t)) \quad (\text{cognitive}) \quad (6.8)$$

$$+ c_2 \mathbf{u} \odot (\mathbf{G}_{\text{best}} - \mathbf{X}_i(t)) \quad (\text{social}) \quad (6.9)$$

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \mathbf{V}_i(t+1) \quad (6.10)$$

for iteration $t+1$ of PSO. There are three operations required by this set of equations:

1. scalar multiplication,
2. multiplication of a vector (or matrix) by a scalar, including the scalar -1,
3. and vector/matrix addition.

We can reduce the required operations to only on vectors by noting that we can partition any matrix by its rows, i.e. for $\mathbf{X} \in \times_{j=1}^N \Delta^{K-1}$ write

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ \vdots \\ \mathbf{x}'_N \end{pmatrix} \quad (6.11)$$

where $\mathbf{x}'_j \in \Delta^{K-1}$ for $j = 1, \dots, N$. Consider scalar $a \in \mathbb{R}$ and vectors $\mathbf{p}, \mathbf{q} \in \Delta^{K-1}$. Then what is the meaning of $a\mathbf{p}$? Or of $\mathbf{p} + \mathbf{q}$? These quantities are generally not elements of Δ^{K-1} (and so this set is not closed under normal definitions of scalar multiplication and vector addition). This is easy to see with a simple example. Let

$K = 3$, $a = -1$, $\mathbf{p} = \left(\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3}\right)'$ and $\mathbf{q} = \left(1 \quad 0 \quad 0\right)'$. Then

$$a\mathbf{p} = \begin{pmatrix} -\frac{1}{3} \\ -\frac{1}{3} \\ -\frac{1}{3} \end{pmatrix} \notin \Delta^{K-1}$$

and

$$\mathbf{p} + \mathbf{q} = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{4}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} \notin \Delta^{K-1}.$$

This is to say that a straight forward application of the PSO update equations will have the particles leaving the search space with every step.

Wong et al. (2015) [94] provide the only publication we have found that attempts to adapt PSO to the mixture experiment space. First, note that they provide an algorithm for generating optimal continuous mixture designs, which is quite a different problem than the one we are addressing, i.e. to generate optimal exact mixture designs. Second, they apply the standard definitions of scalar multiplication and vector addition in the PSO update equations (they also chose to vectorize the matrix candidates). To solve the issue of particles leaving Δ^{K-1} they simply apply a normalization to each particle vector such that the design points within that vector sum to 1 before they evaluate the fitness and update personal and global best positions. Third, they only apply the $\mathbf{g}_{\text{best}}$ topology whereas we have illustrated for optimal design the $\mathbf{l}_{\text{best}}$ communication topology is more effective and making the algorithm robust to entrapment in local optima. While illustrated to generate known optimal designs, we find their approach somewhat unsatisfying: every time a particle moves it leaves the search space. When the particles are initialized, they are not starting in Δ^{K-1} . When the random draws occur to perturb the social and cognitive components in the update equations, these are directions in the Cartesian coordinate

system and thus the equations almost never allow the particles an opportunity to move directly toward the personal or global best positions. In light of these observations we were motivated to attempt to construct a PSO algorithm that relies on a self consistent mathematical system on Δ^{K-1} .

6.2.2 The Aitchison geometry

Simply put, the Aitchison geometry says to forget that $\Delta^{K-1} \subset \mathbb{R}^K$ and to work with the geometry and mathematical properties of Δ^{K-1} itself. The geometry is named after John Aitchison, a Scottish statistician and fellow of the Royal Society of Edinburgh. Aitchison practiced from the 1950s into the 2000s, and he died in 2016. He is known for (other than the geometry) an early book (1957) on the lognormal distribution, work with the log-ratio, and compositional data analysis (CODA).

Aitchison (1982) [1] provides a discussion of the simplex as the sample space for random vectors (i.e. the response) which sum to one. The author discusses basic mathematical operations for these type of data, the use of the Dirichlet distribution for inference, transformations to normality, conditions for independence, and a discussion for how to conduct linear regression with compositional data as the response. The paper indicates that compositional data often arise in geological sampling where, for example, the molecular profile of rocks is measured in proportions. Aitchison (1984) [2] provides a development of a general class of distributions on the simplex. The author provides a construction of a general class of distributions which subsumes the Dirichlet class and logistic-normal classes as special cases. The author highlights that the role of the new distributional class is to help assess the independence hypotheses for given compositional data (which is required for many statistical analyses). Aitchison and Bacon-Shone (1984) [4] provide an entry focusing on experimental design for mixture experiments. In contrast to the aforementioned papers, here the predictor

variables are compositional. The authors discuss the use of the log-contrast model for inference in mixtures and identify specific hypotheses related to the model structure. Aitchison (1999) [3] provides a discussion in Mathematical Geology of the inability to interpret correlation computed on raw compositional data and provides an illustration of transformations that make it possible to do so. It is possible to continue adding such references for discussion here (there are many). Suffice it to say the body of literature supporting CODA focuses on ‘fixing’ issues with applying standard statistical analysis to compositional data. It is easy to find references that discuss distributional theory, central limit theorem variants, principle component analysis, discriminant analysis, regression analysis, and exploratory data analysis for compositional data.

Our take-away from surveying the CODA literature is the following: it is possible to define mathematical operations on this space that ensure the simplex is closed with respect to these operations. We develop these operations in the next subsection. We note, however, to take advantage of this system of mathematics we must change the definition of the design space to be an open set, which we denote Δ_O^{K-1} .

6.2.3 Inducing a vector space on Δ^{K-1}

Euclidean space is the classical and familiar geometry one works with on $\mathbb{R}^K$ when applying the standard definitions of vector addition and multiplication of a scalar by a vector. This enables, as the PSO update equations require, displacement and scaling of vectors.

A vector space (or linear space) is a collection of vectors that may be added together and scaled by real numbers. A vector space must meet 8 axioms which we state in the following definition [79].

Definition 2. *Let $\mathcal{V}$ be a set of vectors and let $\mathcal{F}$ be a field of scalars. Let there be two operations*

1. *(vector addition) $+: \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$ and*

2. (scalar multiplication) $\cdot : \mathcal{F} \times \mathcal{V} \rightarrow \mathcal{V}$.

for which $\mathcal{V}$ is closed. Let $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathcal{V}$, and $a, b \in \mathcal{F}$. If

Axiom 1: (associativity of addition)

$$\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathcal{V}, \mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w},$$

Axiom 2: (commutativity of addition)

$$\forall \mathbf{u}, \mathbf{v} \in \mathcal{V}, \mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u},$$

Axiom 3: (identity element of addition)

$$\forall \mathbf{v} \in \mathcal{V}, \exists \mathbf{0} \in \mathcal{V} \ni: \mathbf{v} + \mathbf{0} = \mathbf{v},$$

Axiom 4: (inverse elements of addition)

$$\forall \mathbf{v} \in \mathcal{V}, \exists -\mathbf{v} \in \mathcal{V} \ni: \mathbf{v} + (-\mathbf{v}) = \mathbf{0},$$

Axiom 5: (compatibility of scalar multiplication with field multiplication)

$$\forall a, b \in \mathcal{F} \wedge \forall \mathbf{v} \in \mathcal{V}, a(b\mathbf{v}) = (ab)\mathbf{v},$$

Axiom 6: (identity element of scalar multiplication)

$$\forall \mathbf{v} \in \mathcal{V}, e\mathbf{v} = \mathbf{v} \text{ where } e \text{ is the multiplicative identity in } \mathcal{F},$$

Axiom 7: (distributivity of scalar multiplication with respect to vector addition)

$$\forall a \in \mathcal{F} \wedge \forall \mathbf{u}, \mathbf{v} \in \mathcal{V}, a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}, \text{ and}$$

Axiom 8: (distributivity of scalar multiplication with respect to field addition)

$$\forall a, b \in \mathcal{F} \wedge \forall \mathbf{v} \in \mathcal{V}, (a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$$

then $\mathcal{V}$ is said to be a vector space with respect to $+$ and $\cdot$.

To induce a vector space on Δ_O^{K-1} we define vector addition and scalar multiplication as in the Aitchison Geometry [75]. Let

$$\Delta_O^{K-1} := \left\{ \mathbf{x} \in \mathbb{R}^K \mid \mathbf{1}'_K \mathbf{x} = 1, 0 < \mathbf{e}'_i \mathbf{x} < 1 \text{ for } i = 1, \dots, K \right\} \quad (6.12)$$

be the open standard K -simplex. Let $\mathbf{p}, \mathbf{q} \in \Delta_O^{K-1}$ and $\lambda \in \mathbb{R}$. Let $\odot$ denote the Hadamard product, or power (as appropriate), thus

$$\mathbf{p} \odot \mathbf{q} = \{p_i q_i\} \quad (6.13)$$

and

$$\mathbf{p}^{\odot \lambda} = \{p_i^\lambda\}. \quad (6.14)$$

Define vector addition on Δ_O^K as

$$\mathbf{p} \oplus \mathbf{q} := \frac{\mathbf{p} \odot \mathbf{q}}{\mathbf{1}'_K(\mathbf{p} \odot \mathbf{q})} \quad (6.15)$$

and define scalar multiplication as

$$\lambda \otimes \mathbf{p} := \frac{\mathbf{p}^{\odot \lambda}}{\mathbf{1}'_K(\mathbf{p}^{\odot \lambda})}. \quad (6.16)$$

In the next section we prove that these definitions induce a vector space.

6.2.3.1 Proof that $\oplus$ and $\otimes$ induce a vector space on Δ_O^{K-1} over $\mathbb{R}$ The proof is accomplished by proving that $\Delta_O^{(K-1)}$ satisfies each of the axioms of a vector space over field $\mathbb{R}$.

Axiom 1. *Associativity of Addition:*

$$\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \Delta_O^{K-1}, \mathbf{u} \oplus (\mathbf{v} \oplus \mathbf{w}) = (\mathbf{u} \oplus \mathbf{v}) \oplus \mathbf{w}$$

Proof. Let $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \Delta_O^{K-1}$ be arbitrary but fixed (ABF). Consider the left-hand side

(LHS):

$$\begin{aligned}
 \mathbf{v} \oplus \mathbf{w} &= \frac{\mathbf{v} \odot \mathbf{w}}{\mathbf{1}'_K(\mathbf{v} \odot \mathbf{w})} && (\text{defn of } \oplus \text{ on } \Delta_O^{K-1}) \\
 &= \frac{\{v_i w_i\}}{\sum_{j=1}^K v_j w_j} && (\text{defn of } \odot, \text{ multiplication of } \mathbf{1}_K) \\
 &= \left\{ \frac{v_i w_i}{\sum_{j=1}^K v_j w_j} \right\} && (\text{standard scalar multiplication})
 \end{aligned}$$

Further, by similar steps,

$$\begin{aligned}
 \mathbf{u} \oplus (\mathbf{v} \oplus \mathbf{w}) &= \frac{\mathbf{u} \odot (\mathbf{v} \oplus \mathbf{w})}{\mathbf{1}'_K(\mathbf{u} \odot (\mathbf{v} \oplus \mathbf{w}))} \\
 &= \frac{\left\{ \frac{u_i v_i w_i}{\sum_{j=1}^K v_j w_j} \right\}}{\frac{\sum_{k=1}^K u_k v_k w_k}{\sum_{j=1}^K v_j w_j}} \\
 &= \left\{ \frac{u_i v_i w_i}{\sum_{k=1}^K u_k v_k w_k} \right\}
 \end{aligned}$$

Consider the right hand side (RHS):

$$\begin{aligned}
 \mathbf{u} \oplus \mathbf{v} &= \frac{\mathbf{u} \odot \mathbf{v}}{\mathbf{1}'_K(\mathbf{u} \odot \mathbf{v})} && (\text{defn of } \oplus \text{ on } \Delta_O^{K-1}) \\
 &= \frac{\{u_i v_i\}}{\sum_{j=1}^K u_j v_j} && (\text{defn of } \odot, \text{ multiplication of } \mathbf{1}_K) \\
 &= \left\{ \frac{u_i v_i}{\sum_{j=1}^K u_j v_j} \right\} && (\text{standard scalar multiplication})
 \end{aligned}$$

$$\begin{aligned}
(\mathbf{u} \oplus \mathbf{v}) \oplus \mathbf{w} &= \frac{(\mathbf{u} \oplus \mathbf{v}) \odot \mathbf{w}}{\mathbf{1}'_K((\mathbf{u} \oplus \mathbf{v}) \odot \mathbf{w})} \\
&= \frac{\left\{ \frac{u_i v_i w_i}{\sum_{j=1}^K u_j v_j} \right\}}{\frac{\sum_{k=1}^K u_k v_k w_k}{\sum_{j=1}^K u_j v_j}} \\
&= \left\{ \frac{u_i v_i w_i}{\sum_{k=1}^K u_k v_k w_k} \right\}
\end{aligned}$$

Thus LHS = RHS. $\therefore \mathbf{u} \oplus (\mathbf{v} \oplus \mathbf{w}) = (\mathbf{u} \oplus \mathbf{v}) \oplus \mathbf{w}, \forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \Delta_O^{K-1}$ ■

Axiom 2. *Commutativity of Addition:*

$$\forall \mathbf{u}, \mathbf{v} \in \Delta_O^{K-1}, \mathbf{u} \oplus \mathbf{v} = \mathbf{v} \oplus \mathbf{u}$$

Proof. Let $\mathbf{u}, \mathbf{v} \in \Delta_O^{K-1}$ be ABF.

$$\begin{aligned}
\mathbf{u} \oplus \mathbf{v} &= \frac{\mathbf{u} \odot \mathbf{v}}{\mathbf{1}'_K(\mathbf{u} \odot \mathbf{v})} && (\text{defn of } \oplus \text{ on } \Delta_O^{K-1}) \\
&= \frac{\{u_i v_i\}}{\sum_{j=1}^K u_j v_j} && (\text{defn of } \odot, \text{ multiplication of } \mathbf{1}_K) \\
&= \left\{ \frac{u_i v_i}{\sum_{j=1}^K u_j v_j} \right\} && (\text{standard scalar multiplication}) \\
&= \left\{ \frac{v_i u_i}{\sum_{j=1}^K v_j u_j} \right\} && (\text{commutativity of scalar multiplication}) \\
&= \frac{\{v_i u_i\}}{\sum_{j=1}^K v_j u_j} && (\text{standard scalar multiplication}) \\
&= \frac{\mathbf{v} \odot \mathbf{u}}{\mathbf{1}'_K(\mathbf{v} \odot \mathbf{u})} && (\text{defn of } \odot, \text{ multiplication of } \mathbf{1}_K) \\
&= \mathbf{v} \oplus \mathbf{u} && (\text{defn of } \oplus \text{ on } \Delta_O^{K-1})
\end{aligned}$$

$\therefore \mathbf{u} \oplus \mathbf{v} = \mathbf{v} \oplus \mathbf{u}, \forall \mathbf{u}, \mathbf{v} \in \Delta_O^{K-1}$ ■

Axiom 3. *Identity Element of Addition:*

$$\forall \mathbf{v} \in \Delta_O^{K-1}, \exists \mathbf{0} \in \Delta_O^{K-1} \ni: \mathbf{v} \oplus \mathbf{0} = \mathbf{v}$$

Proof. Let $\mathbf{v} \in \Delta_O^{K-1}$ be ABF. Consider the centroid of Δ_O^{K-1} :

$$\mathbf{c} = \left\{ \frac{1}{K} \right\}$$

Note that $\mathbf{c} \in \Delta_O^{K-1}$ by definition because:

1. The dimension of the space $K \in \mathbb{N}$ has bounds $1 \leq K < \infty$. This implies $0 < \frac{1}{K} \leq 1$, and
2. $\mathbf{1}'_K \mathbf{c} = \sum_{i=1}^K \frac{1}{K} = \frac{K}{K} = 1$.

Now consider

$$\begin{aligned}
 \mathbf{v} \oplus \mathbf{c} &= \frac{\mathbf{v} \odot \mathbf{c}}{\mathbf{1}'_K (\mathbf{v} \odot \mathbf{c})} && (\text{defn of } \oplus \text{ on } \Delta_O^{K-1}) \\
 &= \frac{\left\{ v_i \frac{1}{K} \right\}}{\sum_{j=1}^K v_j \frac{1}{K}} && (\text{defn of } \odot, \text{ multiplication of } \mathbf{1}_K) \\
 &= \frac{\left\{ v_i \frac{1}{K} \right\}}{\frac{1}{K} \sum_{j=1}^K v_j} && (\text{moved scalar outside of sum}) \\
 &= \frac{\left\{ v_i \frac{1}{K} \right\}}{\frac{1}{K}} && \left(\mathbf{v} \in \Delta_O^{K-1} \implies \sum_{j=1}^K v_j = \mathbf{1}'_K \mathbf{v} = 1 \right) \\
 &= \left\{ v_i \frac{K}{K} \right\} && (\text{standard scalar multiplication}) \\
 &= \{v_i\} \\
 &= \mathbf{v}
 \end{aligned}$$

which demonstrates that $\mathbf{c}$ is the additive identity for the definition of $\oplus$ on Δ_O^{K-1} . Thus we define

$$\mathbf{0} := \left\{ \frac{1}{K} \right\}$$

$$\therefore \forall \mathbf{v} \in \Delta_O^{K-1}, \exists \mathbf{0} \in \Delta_O^{K-1} \ni: \mathbf{v} \oplus \mathbf{0} = \mathbf{v} \quad \blacksquare$$

Axiom 4. *Inverse Elements of Addition:*

$$\forall \mathbf{v} \in \Delta_O^{K-1}, \exists -\mathbf{v} \in \Delta_O^{K-1} \ni: \mathbf{v} \oplus (-\mathbf{v}) = \mathbf{0}$$

Proof. Let $\mathbf{v} \in \Delta_O^{K-1}$. Let $\lambda = -1 \in \mathbb{R}$. By definition of scalar multiplication on Δ_O^{K-1} we can construct the form of $-\mathbf{v}$:

$$-\mathbf{v} = (-1) \otimes \mathbf{v} = \frac{\mathbf{v}^{\odot -1}}{\mathbf{1}'_K(\mathbf{v}^{\odot -1})} = \frac{\left\{ \frac{1}{v_i} \right\}}{\sum_{i=1}^K \frac{1}{v_i}}$$

Note that this is valid as Δ_O^{K-1} is defined as an open set, i.e., we have $0 < \mathbf{e}'_i \mathbf{p} < 1$ for $i = 1, \dots, K$. Next, by applying the definition of $\oplus$ on Δ_O^{K-1} we can develop the expression for vector subtraction on Δ_O^{K-1} as:

$$\begin{aligned} \mathbf{v} - \mathbf{v} &= \mathbf{v} \oplus ((-1) \otimes \mathbf{v}) = \frac{\mathbf{v} \odot ((-1) \otimes \mathbf{v})}{\mathbf{1}'_K(\mathbf{v} \odot ((-1) \otimes \mathbf{v}))} \\ &= \frac{\left\{ \frac{\frac{v_i}{v_i}}{\sum_{i=1}^K \frac{1}{v_i}} \right\}}{\frac{\sum_{j=1}^K \frac{v_j}{v_j}}{\sum_{i=1}^K \frac{1}{v_i}}} = \frac{\left\{ \frac{1}{\sum_{i=1}^K \frac{1}{v_i}} \right\}}{\frac{K}{\sum_{i=1}^K \frac{1}{v_i}}} \\ &= \left\{ \frac{1}{K} \right\} = \mathbf{0} \end{aligned}$$

$$\therefore \forall \mathbf{v} \in \Delta_O^{K-1}, \exists -\mathbf{v} \in \Delta_O^{K-1} \ni: \mathbf{v} \oplus (-\mathbf{v}) = \mathbf{0} \quad \blacksquare$$

Axiom 5. *Compatibility of Scalar Multiplication with Field Multiplication:*

$$\forall a, b \in \mathbb{R} \wedge \forall \mathbf{v} \in \Delta_O^{K-1}, a \otimes (b \otimes \mathbf{v}) = (ab) \otimes \mathbf{v}$$

Proof. Let $a, b \in \mathbb{R}$ and $\mathbf{v} \in \Delta_O^{K-1}$ be ABF. LHS: First examine $b \otimes \mathbf{v}$. By Definition we have

$$b \otimes \mathbf{v} = \frac{\mathbf{v}^{\odot b}}{\mathbf{1}'_K(\mathbf{v}^{\odot b})} = \frac{\{v_i^b\}}{\sum_{i=1}^K v_i^b}$$

Next, examine $a \otimes (b \otimes \mathbf{v})$, by definition

$$\begin{aligned} a \otimes (b \otimes \mathbf{v}) &= \frac{(b \otimes \mathbf{v})^{\odot a}}{\mathbf{1}'_K((b \otimes \mathbf{v})^{\odot a})} = \frac{\left\{ \left(\frac{v_i^b}{\sum_{i=1}^K v_i^b} \right)^a \right\}}{\sum_{j=1}^n \left(\frac{v_j^b}{\sum_{i=1}^K v_i^b} \right)^a} \\ &= \frac{\left\{ \frac{v_i^{ab}}{(\sum_{i=1}^K v_i^b)^a} \right\}}{\frac{\sum_{j=1}^n v_j^{ab}}{(\sum_{i=1}^K v_i^b)^a}} = \left\{ \frac{v_i^{ab}}{\sum_{i=1}^K v_i^{ab}} \right\} \end{aligned}$$

RHS: Now examine $(ab) \otimes \mathbf{v}$ by definition of scalar multiplication on Δ_O^{K-1} :

$$(ab) \otimes \mathbf{v} = \frac{\mathbf{v}^{\odot ab}}{\mathbf{1}'_K(\mathbf{v}^{\odot ab})} = \frac{\{v_i^{ab}\}}{\sum_{i=1}^K v_i^{ab}} = \left\{ \frac{v_i^{ab}}{\sum_{i=1}^K v_i^{ab}} \right\}$$

Thus LHS = RHS. $\therefore \forall a, b \in \mathbb{R} \wedge \forall \mathbf{v} \in \Delta_O^{K-1}, a \otimes (b \otimes \mathbf{v}) = (ab) \otimes \mathbf{v}$ ■

Axiom 6. *Identity Element of Scalar Multiplication:*

$$\forall \mathbf{v} \in \Delta_O^{K-1}, 1 \otimes \mathbf{v} = \mathbf{v} \text{ where } 1 \text{ is the multiplicative identity in } \mathbb{R}$$

Proof. Let $\mathbf{v} \in \Delta_O^{K-1}$ be ABF. Clearly $1 \in \mathbb{R}$ is the multiplicative identity in $\mathbb{R}$. Now, consider (by definition)

$$\begin{aligned} 1 \otimes \mathbf{v} &= \frac{\mathbf{v}^{\odot 1}}{\mathbf{1}'_K \mathbf{v}^{\odot 1}} \\ &= \frac{\{v_i^1\}}{\sum_{i=1}^n v_i^1} = \{v_i\} = \mathbf{v} \end{aligned}$$

$\therefore \forall \mathbf{v} \in \Delta_O^{K-1}, 1 \otimes \mathbf{v} = \mathbf{v}$ where 1 is the multiplicative identity in $\mathbb{R}$ ■

Axiom 7. *Distributivity of Scalar Multiplication with respect to Vector Addition:*

$$\forall a \in \mathbb{R} \wedge \forall \mathbf{u}, \mathbf{v} \in \Delta_O^{K-1}, a \otimes (\mathbf{u} \oplus \mathbf{v}) = (a \otimes \mathbf{u}) \oplus (a \otimes \mathbf{v})$$

Proof. Let $a \in \mathbb{R}$ and $\mathbf{u}, \mathbf{v} \in \Delta_O^{K-1}$ be ABF.

LHS: First, by definition of $\oplus$ on Δ_O^{K-1}

$$\mathbf{u} \oplus \mathbf{v} = \frac{\mathbf{u} \odot \mathbf{v}}{\mathbf{1}'_K(\mathbf{u} \odot \mathbf{v})} = \frac{\{u_i v_i\}}{\sum_{j=1}^K u_j v_j} = \left\{ \frac{u_i v_i}{\sum_{j=1}^K u_j v_j} \right\}$$

Next, by definition of scalar multiplication on Δ_O^{K-1} we have

$$a \otimes (\mathbf{u} \oplus \mathbf{v}) = \frac{(\mathbf{u} \oplus \mathbf{v})^{\odot a}}{\mathbf{1}'_K(\mathbf{u} \oplus \mathbf{v})^{\odot a}} = \frac{\left\{ \frac{(u_i v_i)^a}{(\sum_{j=1}^K u_j v_j)^a} \right\}}{\frac{\sum_{k=1}^n (u_k v_k)^a}{(\sum_{j=1}^K u_j v_j)^a}} = \left\{ \frac{(u_i v_i)^a}{\sum_{k=1}^K (u_k v_k)^a} \right\}$$

RHS: By definition of scalar multiplication on Δ_O^{K-1} we have

$$a \otimes \mathbf{u} = \left\{ \frac{u_i^a}{\sum_{j=1}^K u_j^a} \right\}, \text{ and } a \otimes \mathbf{v} = \left\{ \frac{v_j^a}{\sum_{j=1}^K v_j^a} \right\}$$

Next, by definition of $\oplus$ on Δ_O^{K-1} we have

$$\begin{aligned} (a \otimes \mathbf{u}) \oplus (a \otimes \mathbf{v}) &= \frac{(a \otimes \mathbf{u}) \odot (a \otimes \mathbf{v})}{\mathbf{1}'_K((a \otimes \mathbf{u}) \odot (a \otimes \mathbf{v}))} = \frac{\left\{ \frac{u_i^a v_i^a}{(\sum_{j=1}^K u_j^a)(\sum_{j=1}^K v_j^a)} \right\}}{\frac{\sum_{k=1}^n u_k^a v_k^a}{(\sum_{j=1}^K u_j^a)(\sum_{j=1}^K v_j^a)}} \\ &= \left\{ \frac{u_i^a v_i^a}{\sum_{k=1}^n u_k^a v_k^a} \right\} = \left\{ \frac{(u_i v_i)^a}{\sum_{k=1}^n (u_k v_k)^a} \right\} \end{aligned}$$

Thus LHS = RHS. $\therefore \forall a \in \mathbb{R} \wedge \forall \mathbf{u}, \mathbf{v} \in \Delta_O^{K-1}, a \otimes (\mathbf{u} \oplus \mathbf{v}) = (a \otimes \mathbf{u}) \oplus (a \otimes \mathbf{v})$ ■

Axiom 8. *Distributivity of Scalar Multiplication with respect to Field Addition:*

$$\forall a, b \in \mathbb{R} \wedge \forall \mathbf{v} \in \Delta_O^{K-1}, (a + b) \otimes \mathbf{v} = (a \otimes \mathbf{v}) \oplus (b \otimes \mathbf{v})$$

Proof. Let $a, b \in \mathbb{R}$ and $\mathbf{v} \in \Delta_O^{K-1}$ be ABF. LHS: By definition of scalar multiplication on Δ_O^{K-1} we have

$$(a + b) \otimes \mathbf{v} = \frac{\mathbf{v}^{\odot(a+b)}}{\mathbf{1}'_K(\mathbf{v}^{\odot(a+b)})} = \frac{\{v_i^{a+b}\}}{\sum_{j=1}^k v_i^{a+b}} = \left\{ \frac{v_i^{a+b}}{\sum_{j=1}^k v_i^{a+b}} \right\}$$

RHS:

$$a \otimes \mathbf{v} = \left\{ \frac{v_i^a}{\sum_{j=1}^K v_j^a} \right\}, \text{ and } b \otimes \mathbf{v} = \left\{ \frac{v_j^b}{\sum_{j=1}^K v_j^b} \right\}$$

Next, by definition of $\oplus$ on Δ_O^{K-1} we have

$$\begin{aligned} (a \otimes \mathbf{v}) \oplus (b \otimes \mathbf{v}) &= \frac{(a \otimes \mathbf{v}) \odot (b \otimes \mathbf{v})}{\mathbf{1}'_K((a\mathbf{v}) \odot (b\mathbf{v}))} = \frac{\left\{ \frac{v_i^a v_i^b}{(\sum_{j=1}^K v_j^a)(\sum_{j=1}^K v_j^b)} \right\}}{\frac{\sum_{k=1}^n v_k^a v_k^b}{(\sum_{j=1}^K v_j^a)(\sum_{j=1}^K v_j^b)}} \\ &= \left\{ \frac{v_i^{a+b}}{\sum_{k=1}^n v_k^{a+b}} \right\} \end{aligned}$$

Thus LHS = RHS. $\therefore \forall a, b \in \mathbb{R} \wedge \forall \mathbf{v} \in \Delta_O^{K-1}, (a + b) \otimes \mathbf{v} = (a \otimes \mathbf{v}) \oplus (b \otimes \mathbf{v})$ ■

Thus, demonstrating that Δ_O^{K-1} satisfies Axioms 1-8. Therefore, we have established that Δ_O^{K-1} is a vector space under the prescribed definitions of vector addition and scalar multiplication.

The implication of this result relative to PSO is that we have an algebraic system of mathematics on Δ_O^{K-1} within which displacing and scaling vectors makes sense, i.e. we have the geometric operations required by the PSO update equations.

6.3 CprodSimplex-PSO: A new PSO algorithm to optimize matrix valued functions on the Cartesian product of standard K -simplices

The optimal design under design criterion F for mixture experiments is defined as

$$\mathbf{X}^* := \underset{\mathbf{X} \in \times_{j=1}^N \Delta^{K-1}}{\operatorname{argmin}} F(\mathbf{X}). \quad (6.17)$$

Recalling our description of PSO algorithms in Chapter 1, we propose that a solution to adapting PSO to this problem requires us to address the following technical issues.

1. (velocity clamping) A maximum velocity is placed on the particles so that they are encouraged to step through the search space and converge to a consensus value.

2. (swarm initialization) Random draw of S candidate matrices on space $\times_{j=1}^N \Delta^{K-1}$. Also random initialization of the particle velocities.
3. (particle confinement) If a particle leaves $\times_{j=1}^N \Delta^{K-1}$, it should be mapped back to the boundary via some concept of shortest distance.
4. (PSO update equations) Based on current knowledge of personal, local, and global positions, the particles velocities should adapt to encourage them to visit those positions subject to random perturbations.

Solutions to these four sub problems are formulated in the next subsections.

6.3.1 CprodSimplex-PSO: velocity clamping

In PSO for searches on the hypercube $\mathcal{X}$, the maximum allowable velocity was set to the half distance across one dimension parallel to the axes. For cubes, this made the maximum velocity just a bit smaller than the half of the maximum distance on the cube (vertex to vertex). We implement something similar on the simplex.

It turns out that the fact that Δ^{K-1} is a vector space can be extended to a normed vector space, and then to a metric space. The norm and distance on Δ^{K-1} are given in [75]. We repeat them here as definitions.

Definition 3. *Norm of $\mathbf{p} \in \Delta_O^{K-1}$,*

$$\|\mathbf{p}\| = \sqrt{\frac{1}{2K} \sum_{i=1}^K \sum_{j=1}^K \left(\ln \frac{p_i}{p_j} \right)^2}. \quad (6.18)$$

Definition 4. *Distance between $\mathbf{p}, \mathbf{q} \in \Delta_O^{K-1}$,*

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\frac{1}{2K} \sum_{i=1}^K \sum_{j=1}^K \left(\ln \frac{p_i}{p_j} - \ln \frac{q_i}{q_j} \right)^2}. \quad (6.19)$$

For a search we compute the maximum distance from the centroid to a vertex, let this distance be $d_{\max}$ and we clamp velocities by $1/2$. Thus, the velocity clamp parameter is set to $v_{\max} = \frac{d_{\max}}{2}$. At particle initialization or velocity updating we check the ij th particle's velocity

$$\|\mathbf{v}'_{ij}\| > v_{\max} \quad (6.20)$$

and if true we set

$$\mathbf{v}'_{ij} = \frac{v_{\max}}{\|\mathbf{v}'_{ij}\|} \otimes \mathbf{v}'_{ij}. \quad (6.21)$$

This subroutine will be referred to as: `clampVelocity`.

6.3.2 CprodSimplex-PSO: swarm initialization

In PSO for optimization of functions on the hypercube $\mathbf{X}^N$, matrix particles were randomly drawn from a uniform distribution of dimension NK . A natural extension to the Beta distribution (and therefore the uniform distribution) for K observations is the Dirichlet distribution which has probability density function [42]

$$p(\mathbf{x}'|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K x_i^{\alpha_i-1}, \quad \text{for } x_i \geq 0 \forall i, \text{ and } \mathbf{x}'\mathbf{1}_K = 1. \quad (6.22)$$

We note that the support of the Dirichlet distribution is Δ^{K-1} . However, as this is a continuous distribution, we believe that there is no issue defining the support as Δ_O^{K-1} as the boundary is a set of measure 0. We will denote the Dirichlet distribution as $\text{Dir}(\mathbf{x}'|\boldsymbol{\alpha})$. Further, note that setting $\boldsymbol{\alpha} = \mathbf{1}_K$ induces a uniform Dirichlet distribution over Δ_O^{K-1} . Random points can be drawn in the design space as

$$\mathbf{x}' \sim \text{Dir}(\mathbf{x}'|\boldsymbol{\alpha}). \quad (6.23)$$

A full $N \times K$ random design matrix (i.e. a particle from PSO's perspective) can be drawn from the *product Dirichlet* distribution:

$$\mathbf{X} \sim \prod_{j=1}^N \text{Dir}(\mathbf{x}' | \boldsymbol{\alpha}). \quad (6.24)$$

Regarding initializing particle velocities, we propose a similar structure: draws from the product Dirichlet. Therefore, we propose Algorithm 6.4 in order to initialize a set of S random design matrices for mixture experiments.

Algorithm 6.4: CprodSimplex-PSO: random uniform swarm initialization of positions.

- 1: **Input:** N, K, S .
- 2: **Output:** $\{\mathbf{X}_i\}_{i=1}^S$: a set of S random mixture designs on $\times_{j=1}^N \Delta^{K-1}$.
- 3: **for each** $i = 1, \dots, S$ **do**
- 4: $\mathbf{X}_i \sim \prod_{j=1}^N \text{Dir}(\mathbf{x}' | \mathbf{1}_K)$
- 5: $\mathbf{V}_i \sim \text{clampVelocity}(\prod_{j=1}^N \text{Dir}(\mathbf{x}' | \mathbf{1}_K))$
- 6: **endfor**
- 7: **return**

6.3.3 CprodSimplex-PSO: particle confinement

The space Δ_O^{K-1} is closed under $\oplus$ and $\otimes$ for all real numbers $\lambda \in \mathbb{R}$. Therefore, as the matrix particles search $\times_{j=1}^N \Delta^{K-1}$, the rows of the candidate matrices will never leave Δ^{K-1} . Therefore, a confinement subroutine is not needed if the design space is the full open simplex Δ_O^{K-1} . In this way, the Aitchison geometry seems to be an elegant approach to handling the constraint that design vectors must sum to 1.

6.3.4 CprodSimplex-PSO: PSO update equations

For searches on the hypercube $\mathcal{X}^N$, the rows and columns of a candidate matrix $\mathbf{X}$ could be considered linearly independent (i.e. the coordinates in the Cartesian system are orthogonal). This gave rise to the velocity and position update equations stated in Eq.s 6.7 - 6.10. The random perturbations inside the velocity update equations were implemented on each element of $\mathbf{X}$.

In mixture experiments the K columns are not independent (as they are required to sum to 1) but the N distinct design points are. Therefore, we propose an adaptation of the PSO update equations which honors the row-wise independence and uses the Aitchison geometry to update the velocity and position of a candidate matrix on a row-wise basis. Our proposed algorithm for the PSO update equations for PSO searches in mixture experiments is shown in Alg. 6.5 as a function of iteration index t and for particle i : where $U \sim \text{Uniform}(0, 1)$, and we set ω, c_1 , and c_2 to the values

Algorithm 6.5: CprodSimplex-PSO: PSO update equations for particle i .

```

1: Input:  $\mathbf{X}_i, \mathbf{V}_i, \mathbf{P}_{\text{best}, i}, \mathbf{G}_{\text{best}}$  .
2: Output: Updated particle velocities and positions in  $\times_{j=1}^N \Delta^{K-1}$ .
3: for each  $j = 1, \dots, N$  do
4:    $\mathbf{i}'_{ij}(t+1) = \omega \otimes \mathbf{v}'_{ij}(t)$  (inertia)
5:    $\mathbf{c}'_{ij}(t+1) = (c_1 * U) \otimes (\mathbf{p}'_{\text{best}, ij} \oplus (-1) \otimes \mathbf{x}'_{ij}(t))$  (cognitive)
6:    $\mathbf{s}'_{ij}(t+1) = (c_2 * U) \otimes (\mathbf{g}'_{\text{best}, j} \oplus (-1) \otimes \mathbf{x}'_{ij}(t))$  (social)
7:    $\mathbf{v}'_{ij}(t+1) = \text{clampVelocity}(\mathbf{i}'_{ij}(t+1) \oplus \mathbf{c}'_{ij}(t+1) \oplus \mathbf{s}'_{ij}(t+1))$ 
8:    $\mathbf{x}'_{ij}(t+1) = \mathbf{x}'_{ij}(t) \oplus \mathbf{v}'_{ij}(t)$ 
9: endfor
10: return
```

recommended in the literature [29]. We recognize that these values were derived for

searches on the hypercube $\mathcal{X}$ and are guaranteed to ensure particle convergence on that space. Provided we do not observe swarm explosion in this new algorithm, these values of the constants are reasonable. Attempting to find optimal values of these constants for searches on $\times_{j=1}^N \Delta^{K-1}$ may be the subject of future research.

6.3.5 Discussion

With those aspects defined in the previous sections all the technical aspects of a PSO for implementing optimization searches on $\times_{j=1}^N \Delta^{K-1}$ have been addressed. In a subsequent section, we apply the new algorithm to optimal design problems with known solutions and illustrate the degree of success of our proposed algorithm.

6.4 Linear models for mixture experiments

Recall from Chapter 2 that one of the inputs to the optimal design search is the form of the model we want the design to support. Linear models for mixture experiments have a slightly different structure than standard linear regression models. This is due to the sum to 1 constraint on the design points. A major implication is that the linear model in mixture experiments cannot contain the intercept term in order to uniquely estimate the β s. Further, cross-products linear terms with squared terms should also be excluded [45]. This can be seen if we investigate a squared term

$$x_i^2 = x_i \left(1 - \sum_{j=1}^K x_j - x_i \right) = x_i - \sum_{j=1}^K x_j x_i - x_i^2 \quad (6.25)$$

which illustrates that a squared mixture component is a linear combination of that component and the cross-products with the $K - 1$ other mixture components. Scheffé (1958) [81] gives a detailed discussion of these issues and proposes a set of linear models that are referred to as *Scheffé polynomials*. We list some of these linear

models that we will apply our algorithm to generate optimal designs in the next subsections. The models stated below are referenced in [14, 45, 81]

6.4.1 First-order Scheffé model

$$E(y) = \sum_{i=1}^K \beta_i x_i \quad (6.26)$$

6.4.2 Second-order Scheffé model

$$E(y) = \sum_{i=1}^K \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \beta_{ij} x_i x_j \quad (6.27)$$

6.4.3 Special cubic model

$$E(y) = \sum_{i=1}^K \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \beta_{ij} x_i x_j + \sum_{i=1}^{K-2} \sum_{j=i+1}^{K-1} \sum_{k=j+1}^K \beta_{ijk} x_i x_j x_k \quad (6.28)$$

6.4.4 Full cubic model

$$\begin{aligned} E(y) = & \sum_{i=1}^K \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \beta_{ij} x_i x_j + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \gamma_{ij} x_i x_j (x_i - x_j) \\ & + \sum_{i=1}^{K-2} \sum_{j=i+1}^{K-1} \sum_{k=j+1}^K \beta_{ijk} x_i x_j x_k \end{aligned} \quad (6.29)$$

6.4.5 Special quartic model

$$E(y) = \sum_{i=1}^K \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \beta_{ij} x_i x_j + \sum_{i=1}^{K-2} \sum_{j=i+1}^{K-1} \sum_{k=j+1}^K \nu_{ijk} x_i^2 x_j x_k \quad (6.30)$$

6.4.6 Full quartic model

$$\begin{aligned}
E(y) = & \sum_{i=1}^K \beta_i x_i + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \beta_{ij} x_i x_j + \sum_{i=1}^{K-1} \sum_{j=i+1}^K \delta_{ij} x_i x_j (x_i - x_j) + \\
& \sum_{i=1}^{K-1} \sum_{j=i+1}^K \eta_{ij} x_i x_j (x_i - x_j)^2 + \sum_{i=1}^{K-2} \sum_{j=i+1}^{K-1} \sum_{k=j+1}^K \nu_{ijk} x_i^2 x_j x_k
\end{aligned} \tag{6.31}$$

6.5 Optimality criteria for mixture designs

We will focus on the D and IV -optimality. The D -criterion is computed as stated in Chapter 2 (i.e. it is simply the determinant of the model information matrix). The IV -criterion, however, requires some adjustment to compute it over $\times_{j=1}^N \Delta^{K-1}$. Specifically, the limits of integration change. Recall that

$$IV(\mathbf{X}) = \frac{1}{V} \text{tr} \left\{ (\mathbf{F}'\mathbf{F})^{-1} \mathbf{W} \right\} \tag{6.32}$$

where $V = \int_{\Delta^{K-1}} \mathbf{d}\mathbf{x}'$ is the volume of $\mathcal{X}$ and

$$\mathbf{W} = \int_{\Delta^{K-1}} \mathbf{f}(\mathbf{x}') \mathbf{f}'(\mathbf{x}') \mathbf{d}\mathbf{x}' \tag{6.33}$$

is the *region moments* matrix [44, 68]. The volume has a simple formula if the integration is over the full simplex [45]

$$V = \int_{\Delta^{K-1}} \mathbf{d}\mathbf{x}' = \frac{1}{\Gamma(K)}. \tag{6.34}$$

The moments matrix integration limits have the following form

$$\begin{aligned}\mathbf{W} &= \int_{\Delta^{K-1}} \mathbf{f}(\mathbf{x}')\mathbf{f}'(\mathbf{x}')d\mathbf{x}' \\ &= \int_0^1 \cdots \int_0^{1-\sum_{i=1}^{K-2} x_i} \int_0^{1-\sum_{i=1}^{K-1} x_i} \mathbf{f}(\mathbf{x}')\mathbf{f}'(\mathbf{x}')dx_K dx_{K-1} \cdots dx_1\end{aligned}\quad (6.35)$$

We implemented these calculations in Sage (an open source analogue to Mathematica) to produce the region moment matrices for the models in the previous section [89].

6.6 Validating CprodSimplex-PSO

Our approach to validating the CprodSimplex-PSO algorithm is to apply it to optimal design problems with known solutions. We start with simplex centroid designs. The structure of each study is as follows:

1. Each PSO search will use $S = 150$ particle matrices.
2. We ran $n_{\text{run}} = 210$ PSO searches per design problem (parallelized on 7 cores).
3. Maximum number of iterations is 5000, maximum stagnation is 100 iterations, and a relative tolerance stopping criterion of 3×10^{-13} was used (contingent upon a non-zero change in $f_{\mathbf{G}_{\text{best}}}$).

The distribution of the global best solutions, the best solution found, and the CPU run time will be reported. A comparison of the CprodSimplex-PSO design to the known optimal design will be given.

Note that in the results that follow we will report the D -criterion on the scale of Borkowski (2003) [13] using formula

$$D(\mathbf{X}) = \frac{100}{N} |\mathbf{F}'\mathbf{F}|^{\frac{1}{p}}. \quad (6.36)$$

On this scale a larger score implies a better design (i.e. we are attempting to maximize the design information matrix).

The scale of the IV -criterion, however, will be consistent with Eq. 6.32, and on this scale, a smaller score implies a better design (i.e. we are attempting to minimize the average prediction variance).

6.6.1 (D -criterion) Generating simplex-lattice designs: CprodSimplex-PSO searches on the first and second-order Scheffé models

Classical design in mixture experiments focused on simple coverings of Δ^{K-1} . A simple approach was to place N points in a regular grid, or lattice, over the simplex. Borkowski (2018) [14] gives the definition of a simplex-lattice design:

Definition 5. A $\{K, m\}$ **simplex-lattice design** (SLD) for a K component mixture consists of all points generated by the following rule. Form all possible K -tuples $(x_1, \dots, x_K)$ such that

1. $\sum_{i=1}^K x_i = 1$, and
2. each $x_i \in \{0, \frac{1}{m}, \frac{2}{m}, \dots, \frac{m-1}{m}, 1\}$.

The number of design points is

$$N = \binom{m+K-1}{m}.$$

We present a graphic illustrating SLDs for $K = 3$ and $m = 1, 2, 3, 5, 7, 9$ in Figure 6.1. The ternary diagrams were generated using the **ggtern** package in the R statistical programming language [46, 76]. Kiefer (1961) [59] established that, for continuous designs, SLDs $\{K, 1\}$ and $\{K, 2\}$ are D -optimal minimal support designs for the first and second-order Scheffé model and they have $N^* = p$ unique design

points with p representing the number of parameters.. Goos et al. [45] state that exact D -optimal designs for mixture experiments use the same N^* distinct design points as the continuous designs and if $N > p$ then these points will be replicated, with equal replication occurring if $N = \alpha p$ for some $\alpha \in \mathbb{N}$. Therefore, we can employ CprodSimplex-PSO on searches for D -optimal designs for the first and second-order model and see if the SLDs are returned.

Run time for the first 3 sets was negligible. The $K = 4$ second-order model took approximately 3 minutes to finish all $n_{\text{run}} = 210$ PSO searches.

6.6.1.1 $K = 3$ first order model For this search $p = 3$ so we set $N = 3$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.2. In this case CprodSimplex-PSO found the expected $\{3, 1\}$ design in all of the $n_{\text{run}} = 210$ searches, and, did it quickly with a median number of iterations of 24.

6.6.1.2 $K = 3$ second order model For this search $p = 6$ so we set $N = 6$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.3. In this case CprodSimplex-PSO found the expected $\{3, 2\}$ design in all of the $n_{\text{run}} = 210$ searches, and, did it quickly with a median number of iterations of 170.

6.6.1.3 $K = 4$ first order model For this search $p = 4$ so we set $N = 4$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.4. In this case CprodSimplex-PSO found the expected $\{4, 1\}$ design in all of the $n_{\text{run}} = 210$ searches, and, did it quickly with a median number of iterations of 35.

6.6.1.4 $K = 4$ second order model For this search $p = 10$ so we set $N = 10$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.5. In this case CprodSimplex-PSO found the expected $\{4, 2\}$ design in all of the $n_{\text{run}} = 210$

searches, and, did it quickly with a median number of iterations of 320.

6.6.2 (D -criterion) Generating simplex-centroid designs: CprodSimplex-PSO searches on the special cubic model

Scheffé (1963) published a paper on generating Simplex-centroid designs (SCDs) as an alternative to SLDs [14, 82]. SCDs ensure that design points are placed at the centers of planes and hyperplanes. Borkowski (2018) [14] gives the definition of a simplex-centroid design:

Definition 6. *To construct a K component **simplex-centroid design** (SCD), generate*

1. *All K permutations of the pure mixture components, i.e. $(1, 0, \dots, 0, 0)$,*
2. *All $\binom{K}{2}$ permutations of the binary mixture components, i.e. $(\frac{1}{2}, \frac{1}{2}, 0, \dots, 0)$,*
3. *All $\binom{K}{3}$ permutations of the ternary mixture components, i.e. $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, \dots, 0)$,*
4. *Continue taking all $\binom{K}{k}$ permutations of the k -ary mixture components, i.e. $(\frac{1}{k}, \dots, \frac{1}{k}, 0, \dots, 0)$,*
5. *When $k = K$, the last mixture component is the centroid of Δ^{K-1} , i.e. $(\frac{1}{K}, \dots, \frac{1}{K})$.*

The number of design points is

$$\sum_{i=1}^K \binom{K}{i} = 2^K - 1.$$

Unlike the SLD there is only one SCD per K [45]. We show the SCD for 3 factors in Figure 6.6. Uranisi (1964) [91] demonstrated that a specific fraction of the SCD (for all K) is D -optimal for the special cubic model. Goos et al. [45] define this fraction as:

Definition 7. *The $\{K, 3\}$ -SCD consists of*

1. All K permutations of the pure mixture components, i.e. $(1, 0, \dots, 0, 0)$,
2. All $\binom{K}{2}$ permutations of the binary mixture components, i.e. $(\frac{1}{2}, \frac{1}{2}, 0, \dots, 0)$,
3. All $\binom{K}{3}$ permutations of the ternary mixture components, i.e. $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, \dots, 0)$.

Therefore, we can employ CprodSimplex-PSO on searches for D -optimal designs for the special-cubic model to see if the $\{K, 3\}$ -SCD design is returned. We will investigate the $K = 3, 4$ factor cases. The run time for the suite of $n_{\text{run}} = 210$ PSO searches took approximately 4 and 6 minutes of the $K = 3$ and 4 factor scenarios respectively.

6.6.2.1 $\{3, 3\}$ -SCD: special cubic model For this search $p = 7$ so we set $N = 7$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.7. In this case CprodSimplex-PSO found the expected $\{3, 3\}$ -SCD design in all of the $n_{\text{run}} = 210$ searches, and, did it quickly with a median number of iterations of 225.

6.6.2.2 $\{4, 3\}$ -SCD: special cubic model For this search $p = 14$ so we set $N = 14$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.8. In this case CprodSimplex-PSO found the expected $\{4, 3\}$ -SCD design in all of the $n_{\text{run}} = 210$ searches, and, did it quickly with a median number of iterations of 530. We note that, for this scenario, the optimal design only matches the expected design to 3 decimal places (i.e. 10-thousandths of a proportion).

6.6.3 (D -criterion) Augmented SCDs: CprodSimplex-PSO searches on the full cubic model

Goos et al. (2018) [45] give the D -optimal designs for the full cubic model. These designs include:

1. The K pure components,

2. The $2\binom{K}{2}$ mixtures containing proportions of $\frac{1 \pm \frac{1}{\sqrt{5}}}{2}$, specifically
 - (a) a 0.2764 proportion of one ingredient and
 - (b) a 0.7236 proportion of another component,
3. The $\binom{K}{3}$ ternary mixtures.

Therefore, we can employ CprodSimplex-PSO on searches for D -optimal designs for the full cubic model. We will investigate the $K = 3, 4$ factor cases. The run time for the suite of $n_{\text{run}} = 210$ PSO searches took approximately 3 and 13 minutes of the $K = 3$ and 4 factor scenarios respectively.

6.6.3.1 $K = 3$ full cubic model For this search $p = 10$ so we set $N = 10$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.9. In this case CprodSimplex-PSO found the expected $\{4, 3\}$ -SCD design in all of the $n_{\text{run}} = 210$ searches, and, did it quickly with a median number of iterations of 330.

6.6.3.2 $K = 4$ full cubic model For this search $p = 20$ so we set $N = 20$. The results of the $n_{\text{run}} = 210$ CprodSimplex-PSO searches are contained in Figure 6.10. In this case CprodSimplex-PSO found the expected D -optimal design in all of the $n_{\text{run}} = 210$ searches, and, did it with a median number of iterations of 720. We note that, for this scenario, the optimal design only matches the expected design to 3 decimal places (i.e. 10-thousandths of a proportion).

6.6.4 (IV -criterion) CprodSimplex-PSO searches for IV -optimal mixture designs.

Goos, Jones, and Syafitri (2016) [45] published a paper discussing generating IV -optimal designs for mixture experiments. They highlight that, while the primary objective of mixture experiments is to predict the response at different mixtures, there has (surprisingly) been little work for generating IV -optimal models as they

minimize the average prediction variance over Δ^{K-1} . Their paper gives a review of available literature on this topic, they construct and compare continuous and exact *IV*-optimal designs and compare them to *D*-optimal counterparts, and they highlight the *IV*-scores of *D*-optimal models for different modeling situations.

In the sections that follow, we use CprodSimplex-PSO to generate *IV*-optimal designs for the same K, N scenarios presented in Goos et al. (2016). The reader should note that Goos et al. (2016) produced the *IV*-optimal exact designs using 1,000,000 runs of the mixture coordinate exchange algorithm of Piepel, Cooley and Jones (2005) [74]. We draw attention to the fact that we are running CprodSimplex-PSO only 210 times per design scenario.

6.6.4.1 Second order model, $K = 3$, $N = 6$ runs. It took approximately 1 minute to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.11. All $n_{\text{run}} = 210$ PSO searches apparently found the optimal design. The median number of iterations was 200. We note that while Goos et al. (2016) [45] report that their *IV*-optimal search returned the $\{3, 2\}$ SLD, our design was off by 3 decimal places on the mixture proportions for the binary mixtures (can be seen in the simplex plot if one looks closely). We evaluated the *IV*-criterion for the $\{3, 2\}$ SLD using our implementation of the *IV*-criterion and returned *IV*-score = 0.633333. The *IV*-score for the CprodSimplex-PSO optimal design is *IV*-score = 0.633048 which is slightly smaller (this is a better design).

6.6.4.2 Second order model, $K = 3$, $N = 7$ runs. It took approximately 1 minute to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.12. There appears to be one outstanding solution on the *IV*-criterion graphic, but note the scale. This difference was not large enough to affect the optimal design matrix at a decimal precision that made

a substantive difference. Therefore, all particles found the solution. The median number of iterations is 225. In this case CprodSimplex-PSO found exactly the expected design, an $\{3, 3\}$ -SCD. We note however, that where Goos et al. (2016) [45] report IV -score = 0.4952 for this design, we calculate IV -score = 0.49949.

6.6.4.3 Second order model, $K = 3$, $N = 8$ runs. Note that this is the first design in our validation set that does not relate to some fraction of an SLD or SCD. It took approximately 2 minutes to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.13. CprodSimplex-PSO was able to reproduce the optimal design reported by Goos et al. (2016) [45], all PSO searches apparently found this design. The median number of iterations is 260. Goos et al. report IV -score = 0.4371 for this design. Our calculations yield IV -score = 0.437031 (slightly smaller).

6.6.4.4 Second order model, $K = 3$, $N = 30$ runs. It took approximately 15 minutes to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.14. This is the first scenario where each PSO run did not find the exact same design. Median number of iterations is 680. As reported in Goos et al. (2016) [45] the unique design points correspond to those of the $\{3, 3\}$ -SCD. Goos et al. (2016) report that the replication structure should be:

1. three replicates of the pure components,
2. three replicates of the ternary mixture,
3. six replicates of each of the three binary mixtures.

We report the CprodSimplex-PSO optimal design with IV -score and replication structure in Table 6.1. This design and replication structure matches that reported

by Goos et al. (2016). However, Goos et al. (2016) [45] report an IV -score = 0.1086 whereas we compute IV -score = 0.11229 for the same design.

6.6.4.5 Second order model, $K = 4$, $N = 15$ runs. It took approximately 7 minutes to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.15. The results show that every one of the PSO runs found the same design (i.e. they have the same IV -score). The median number of iterations for these searches is 565. Goos et al. (2016) report an IV -score for their optimal design (based on the coordinate exchange algorithm) of IV -score = 0.3022. Our calculation of the IV -score for their reported design is IV -score = 0.301379. The CprodSimplex-PSO generated optimal design has a calculated score of IV -score = 0.301361, that is, slightly better than both Goos et al. (2016) reported score and our calculation of the score for their reported design. As a result, the CprodSimplex-PSO design differs slightly from theirs. We report a comparison between these two optimal designs in Table 6.2.

6.6.4.6 Special cubic model, $K = 4$, $N = 16$ runs. It took approximately 9 minutes to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.16. The results show that every one of the PSO runs found the same design (i.e. they have the same IV -score). The median number of iterations for these searches is 680. Goos et al. (2016) report that the IV -optimal model for this scenario they constructed consists of

1. each of the 4 pure components,
2. the 6 binary mixtures,
3. the 4 ternary mixtures, and

4. 2 replications of the quaternary mixture.

We report the CprodSimplex-PSO IV -optimal model for this scenario in Table 6.3. It can be seen that this matches the coordinate exchange result.

6.6.4.7 Special cubic model, $K = 4$, $N = 17$ runs. It took approximately 9 minutes to run all $n_{\text{run}} = 210$ PSO searches for this design. The results of the CprodSimplex-PSO searches are contained in Figure 6.17. The results show that every one of the PSO runs found the same design (i.e. they have the same IV -score). The median number of iterations for these searches is 680. We present the CprodSimplex-PSO generated design and the Goos et al. (2016) coordinate exchange generated design in Table 6.4. For all practical purposes the designs are indistinguishable.

6.6.5 Discussion and conclusions

We present a summary table of the above validation exercises conducted with CprodSimplex-PSO in Table 6.8. In short, we consider the new proposed algorithm to work extremely well and thus has been validated.

6.7 Experiments with lower and upper constraints on the mixture proportions

It is common in mixture experiments that the proportions are constrained to an interval of the form [31]

$$0 \leq L_i \leq x_i \leq U_i \leq 1 \quad \text{for } i = 1, \dots, K. \quad (6.37)$$

The constraints define a K -dimensional convex irregular polytope $\diamond^{K-1} \subset \Delta_O^{K-1}$. The space of candidate design matrices is now $\mathbf{X} \in \times_{i=1}^N \diamond^{K-1}$.

To support PSO searches on this space, we identified the need for the following subroutines which construct various geometric summaries of $\diamond^{K-1}$:

1. checking the consistency of $\diamond^{K-1}$ and adjustment to consistency, i.e. an inconsistent $\diamond^{K-1}$ is one at least one L_i or U_i is not a feasible component proportion,
2. calculating the number of D -dimensional surfaces, denoted n_d , of $\diamond^{K-1}$, e.g. n_0 is the number of vertices (points), n_1 is the number of edges (lines), n_2 is the number of 2-D faces, etc.,
3. determining the extreme vertices and the centroid of $\diamond^{K-1}$, and
4. calculating the maximum distance between vertices.

We identified the need for the following additional subroutines of the CprodSimplex-PSO algorithm for generating exact optimal mixture designs for this scenario:

1. velocity clamping,
2. swarm initialization honoring the constraints $0 \leq L_i \leq x_i \leq U_i \leq 1$ for $i = 1, \dots, K$,
3. particle confinement, i.e., if candidate design point $\mathbf{x}' \notin \diamond^{K-1}$ adjust to point $\mathbf{x}^{*'}$ such that it is on the face of $\mathbf{x}^{*'} \in \diamond^{K-1}$ and was moved on a line toward the centroid.

In the next subsections we address all these items, illustrate our approach and provide necessary references.

6.7.1 Functions to compute geometric summaries of $\diamond^{K-1}$

6.7.1.1 Checking and adjusting the consistency of $\diamond^{K-1}$ The polytope $\diamond^{K-1}$ defined by constraints L_i and U_i for $i = 1, \dots, K$ is said to be *consistent* if there exists a mixture component among feasible mixture combinations of x_i such that each x_i achieves the bounds L_i and U_i $i = 1, \dots, K$ (not necessarily simultaneously), i.e. L_i

and U_i are feasible mixture proportions $\forall i$ [31]. If all bounds are not achievable then $\diamond^{K-1}$ is *inconsistent*. Following the description in Wang Ng et al. [69], let

$$L_+ = \sum_{i=1}^K L_i \quad \text{and} \quad U_+ = \sum_{i=1}^K U_i. \quad (6.38)$$

Then $\diamond^{K-1}$ is non-empty if and only if

$$L_+ \leq 1 \leq U_+. \quad (6.39)$$

Methods for detecting inconsistent constraints are discussed in Piepel (1983) and Crosier (1984, 1986), [33,34,73]. Crosier suggests using the L - and U -pseudocomponents. First compute the constraint ranges

$$R_i = U_i - L_i \quad \text{for } i = 1, \dots, K. \quad (6.40)$$

Next, compute the pseudocomponents

$$R_L = 1 - \sum_{i=1}^K L_i \quad \text{and} \quad R_U = \sum_{i=1}^K U_i - 1. \quad (6.41)$$

Then, if for any i $R_i > R_U$ L_i is unattainable and $\diamond^{K-1}$ is inconsistent. Similarly, if for any i $R_i > R_L$ then U_i is unattainable and $\diamond^{K-1}$ is inconsistent. In the case of an inconsistent $\diamond^{K-1}$ Piepel showed that adjusting the constraints to

$$L_i^* = \max(L_i, 1 - U_+ + U_i), \quad \text{and} \quad (6.42)$$

$$U_i^* = \min(U_i, 1 - L_+ + L_i) \quad (6.43)$$

yields an adjusted polytope $\diamond_*^{K-1}$ equivalent to $\diamond^{K-1}$ with no superfluous constraints [31, 73].

6.7.1.2 Calculating the number of D -dimensional surfaces of $\diamond^{K-1}$ Following Cornell (2011) and Crosier (1986) [31, 34] a first step into understanding the geometric properties of $\diamond^{K-1}$ is to enumerate the D -dimensional boundaries of this polytope where the dimension can range from $d = 1$ to $d = K - 2$.

First, compute ranges R_i and the lower and upper pseudocomponents R_L and R_U . Define

$$R_p = \min(R_L, R_U). \quad (6.44)$$

Let $C(K, r) = \binom{K}{r}$ be the number of ranges taken r at a time. Divide the $C(K, r)$ ranges into the following mutually exclusive and exhaustive subsets defined by

$$L(r) := \text{the number of combinations of component ranges such that } \sum R_i < R_p$$

$$E(r) := \text{the number of combinations of component ranges such that } \sum R_i = R_p$$

$$G(r) := \text{the number of combinations of component ranges such that } \sum R_i > R_p.$$

Then the number of vertices (denoted when $d = 0$) is counted as

$$n_0 = K + \sum_{r=1}^K (L(r)(K - 2r) - E(r)(r - 1)). \quad (6.45)$$

And the number of d -dimensional boundaries is counted as

$$\begin{aligned} n_d = & C(K, K - d - 1) + \sum_{r=1}^{K-d-1} L(r)C(K - r, K - r - d - 1) \\ & - \sum_{r=d+1}^q (L(r) + E(r))C(r, r - k - 1). \end{aligned} \quad (6.46)$$

6.7.1.3 Calculating the extreme vertices and centroid of $\diamond^{K-1}$ In this work so far we have implemented the McLean and Anderson (1966) extreme vertices algorithm [31,65]. While functional, this algorithm must be programmed on a case-by-case basis (we currently have cases $K = 3, 4$ implemented. Future work may include porting the XVERT and XVERT1 algorithms into Julia [70, 71]. McLean and Anderson (1966) prescribe a two-step procedure for using the L_i and U_i to generate the extreme vertices of $\diamond^{K-1}$:

1. As in a two-level factorial design, list all possible combinations of the L_i and U_i for $K - 1$ components leaving the value of the remaining component blank. This produces 2^{K-1} combinations. Repeat for each factor, this gives a total of $K \times 2^{K-1}$ combinations. Fill in the blank component by subtracting the sum of the others from 1.
2. Retain only those combinations that yield a valid mixture (i.e. reject any calculated mixture component proportions that are not feasible). The unique values of these valid combinations are the extreme vertices.

Averaging groups of the vertices with a common component will yield face and edge centers. The centroid is generated by averaging all vertices. The result of this calculation is the centroid $\mathbf{c}$ and the vertices $\mathbf{v}_1, \dots, \mathbf{v}_{n_0}$, and additionally, polytope edge and face centroids.

6.7.1.4 Calculating the maximum distance between vertices. This quantity is required to measure the “size” of $\diamond^{K-1}$ (boundary to boundary) in order to define a step size for the particles in the CprodSimplex-PSO algorithm. Let

$$\{\mathbf{v}_1, \dots, \mathbf{v}_{n_0}\} \tag{6.47}$$

be the vertices of $\diamond^{K-1}$. Then compute

$$d_{\max} = \max_{i,j} d(\mathbf{v}_i, \mathbf{v}_j) \quad (6.48)$$

where $i \neq j = 1, \dots, n_0$ and d is given by Eq. 6.19.

6.7.2 Additional subroutines required to support PSO searches on $\diamond^{K-1}$

6.7.2.1 Velocity clamping on $\diamond^{K-1}$ We set the maximum particle velocity to $v_{\max} = \frac{d_{\max}}{4}$. At particle initialization or velocity updating we check if the ij th particle velocity satisfies

$$\|\mathbf{v}'_{ij}\| > v_{\max} \quad (6.49)$$

and, if true, we set

$$\mathbf{v}'_{ij} = \frac{v_{\max}}{\|\mathbf{v}'_{ij}\|} \otimes \mathbf{v}'_{ij}. \quad (6.50)$$

This subroutine will be referred to as: `clampVelocityPolytope`.

6.7.2.2 The truncated Dirichlet distribution: initializing a swarm of random design matrices on $\diamond^{K-1}$ Recall that for searches on the unconstrained $\times_{i=1}^N \Delta^{K-1}$ we drew design points

$$\mathbf{x}' \sim \text{Dir}(\mathbf{x}' | \mathbf{1}_K)$$

and design matrices from

$$\mathbf{X} \sim \prod_{j=1}^N \text{Dir}(\mathbf{x}' | \mathbf{1}_K).$$

Now, given constraints

$$0 \leq L_i \leq x_i \leq U_i \leq 1$$

for $i = 1, \dots, K$ we must draw N random vectors $\mathbf{x}'$ such that the i th element conforms to these constraints. This suggests a truncated Dirichlet distribution (TDir). Group

the lower and upper bounds into vectors $\mathbf{l}$ and $\mathbf{u}$ respectively.

Some proposed procedures for drawing from a truncated Dirichlet distribution are summarized in Wang Ng (2011) [69] and also described in Fang et al. (2000) [41]. The first proposed method is to implement an inverse CDF procedure based on the probability integral transform [22] and a recursive factorization of the Dirichlet distribution into one marginal and the conditional distributions. We now summarize the procedure.

The sampling procedure relies on a factorization of the joint distribution. Here, $f(\mathbf{x}')$ will be used to denote the joint PDF of $\text{TDDir}(\mathbf{x}'|\boldsymbol{\alpha})$ distribution where TDir is an abbreviation for the truncated Dirichlet distribution. Noting that the last variate x_K is determined if the first $K - 1$ x_i for $i = 1, \dots, K - 1$ are realized, it is useful to consider the equivalent random vector $\mathbf{x}'_{-K}$, which has the x_K entry removed, and which has distribution $f(\mathbf{x}'_{-K}) = f(\mathbf{x}')$. Now, by applying the definition of conditional probability [7]

$$f(\mathbf{x}'_{-K}) = \left\{ \prod_{k=1}^{K-2} f(x_k | x_{k+1}, x_{k+2}, \dots, x_{K-1}) \right\} \times f(x_{K-1}). \quad (6.51)$$

Eq. 6.51 shows us that we must identify the conditional and marginal distributions relative to $\text{TDDir}(\mathbf{x}'|\boldsymbol{\alpha})$.

First we state the PDF of the *truncated Beta* distribution (TBeta) and an inverse CDF method for simulating from it. Let L and U be scalar constraints for one dimension (these are the truncation values). Then the notation $X \sim$

$\text{TBeta}(\alpha_1, \alpha_2; L, U)$ represents the truncated Beta distribution which has CDF

$$F_{\text{TB}}(x) = \begin{cases} 0, & \text{if } x < L \\ \frac{I_x(\alpha_1, \alpha_2) - I_L(\alpha_1, \alpha_2)}{I_U(\alpha_1, \alpha_2) - I_L(\alpha_1, \alpha_2)}, & \text{if } L \leq x \leq U \\ 1, & \text{if } U \leq x \end{cases} \quad (6.52)$$

where

$$I_x(\alpha_1, \alpha_2) = \frac{1}{B(\alpha_1, \alpha_2)} \int_0^x t^{\alpha_1-1} (1-t)^{\alpha_2-1} dt$$

is the CDF of the $\text{Beta}(\alpha_1, \alpha_2)$ distribution. Its possible to simulate from the truncated Beta distribution via the inverse CDF method as follows. Let $U \sim U(0, 1)$ and set $U = F_{\text{TB}(X)}$. Solving for x yields

$$X = I_x^{-1}(I_L + U(I_U - I_L)) \quad (6.53)$$

which are the desired simulants.

Next we describe the marginal distribution. We repeat Theorem 7.1 in Wang Ng (2011) [69].

Theorem 4. *If $\mathbf{x} \sim \text{Tdir}(\mathbf{x}|\boldsymbol{\alpha}; \mathbf{l}, \mathbf{u})$ then the marginal distribution*

$$x_{K-1} \sim \text{TBeta}(\alpha_{K-1}, \alpha_+ - \alpha_{K-1}; \xi_{K-1}, \eta_{K-1})$$

where $\alpha_+ = \sum_{i=1}^K \alpha_i$, and TBeta stands for the truncated Beta distribution with lower and upper bounds of

$$\xi_{K-1} = \max(L_{K-1}, 1 - U_+ + U_{K-1})$$

$$\eta_{K-1} = \min(U_{K-1}, 1 - L_+ + L_{K-1}).$$

And similarly Theorem 7.2 which gives the conditional distributions relative to the Dirichlet distribution.

Theorem 5. *If $\mathbf{x}'_{-K} \sim \text{TDDir}(\alpha_{-K}; \alpha_K, \mathbf{1}, \mathbf{u})$ then for $k = K - 2, K - 3, \dots, 1$*

$$\left(\frac{x_k}{1 - \sum_{j=k+1}^{K-1} x_j} \middle| x_{k+1}, \dots, x_{K-1} \right) \sim \text{TBeta} \left(\alpha_k, \alpha_+ - \sum_{j=k}^{K-1} \alpha_j; \xi_k, \eta_k \right)$$

where

$$\begin{aligned} \xi_k &= \max \left(\frac{L_k}{1 - \sum_{j=k+1}^{K-1} x_j}, 1 - \frac{U_+ - \sum_{j=k}^{K-1} U_j}{1 - \sum_{j=k+1}^{K-1} x_j} \right) \\ \eta_k &= \min \left(\frac{U_k}{1 - \sum_{j=k+1}^{K-1} x_j}, 1 - \frac{L_+ - \sum_{j=k}^{K-1} L_j}{1 - \sum_{j=k+1}^{K-1} x_j} \right). \end{aligned}$$

Theorems 4 and 5 along with the CDF of TB in Eq. 6.52 and the simulants in Eq. 6.53 give us the tools to sample from the joint as factored in Eq. 6.51. This is Theorem 7.3 in Wang Ng (2011) [69] which we repeat below.

Theorem 6. *If $\mathbf{x}' \sim \text{TDDir}(\boldsymbol{\alpha}; \mathbf{1}, \mathbf{u})$ on $\diamond^{K-1}$ and $u_1, \dots, u_{K-1} \stackrel{iid}{\sim} U(0, 1)$, then $\mathbf{x}'$ has the following stochastic representation:*

$$\begin{cases} x_{K-1} = F_{K-1}^{-1}(u_{K-1}), \\ x_k = \left(1 - \sum_{j=k+1}^{K-1} x_j\right) F_k^{-1}(u_k), \quad \text{for } k = K - 2, \dots, 1, \\ x_K = 1 - \sum_{k=1}^{K-1} x_k \end{cases}$$

where F_k^{-1} is the inverse CDF of a

$$\text{TBeta} \left(\alpha_k, \alpha_+ - \sum_{j=k}^{K-1} \alpha_j; \xi_k, \eta_k \right)$$

where ξ_k and η_k are given in 5 for $k = K - 1, K - 2, \dots, 1$ and the sum $\sum_{j=k+1}^{K-1}$ is defined

as 0 if $k + 1 > K - 1$.

Theorem 6 gives us a way to sample from a truncated Dirichlet distribution and hence form a product truncated Dirichlet distribution and which can be applied to initialize candidate matrices $\mathbf{X} \in \times_{i=1}^K \diamond^{K-1}$.

As an alternate approach the above mentioned authors also propose a Gibbs sampler. We repeat their procedure here. Let $\mathbf{x}_{-k}$ be the Dirichlet random vector with the k th element removed. The Gibbs sampler requires specification of the full conditional density $f(x_k|\mathbf{x}_{-k})$ for $k = 1, \dots, K$. It can be verified that

$$f(x_k|\mathbf{x}_{-k}) \propto x_k^{\alpha_k-1} (x_k^* - x_k)^{\alpha_K-1}$$

with support

$$\max(L_k, x_k^* - U_K) \leq x_k \leq \min(U_k, x_k^* - a_K)$$

where

$$x_i^* = x_K + x_k = 1 - \mathbf{1}_{K-1}' \mathbf{x}_{-k}.$$

Setting

$$y_k = \frac{x_i}{x_i^*}, \quad k = 1, \dots, K - 1,$$

it can be shown that

$$y_k|\mathbf{x}_{-k} \sim \text{TBeta}(\alpha_k, \alpha_K; L_k^*, U_k^*)$$

where $L_k^* = \max\left(\frac{L_k}{x_k^*}, 1 - \frac{U_K}{x_k^*}\right)$, $U_k^* = \min\left(\frac{U_k}{x_k^*}, 1 - \frac{L_K}{x_k^*}\right)$. Given y_k computing

$$x_k = x_k^* y_k$$

completes the Gibbs sampler and produces the desired simulant.

We coded and implemented both procedures from the ground up in both R and

Julia. We observed that, contrary to the authors claim that setting $\alpha = \mathbf{1}$ will produce a uniform distribution on $\diamond^{K-1}$, the probability integral transform favored slightly the first variate picked as the marginal distribution and the resulting joint sample was not uniform. We assure the reader that we did our due diligence in attempting to find errors in the coding (I programmed the approach 4 times over in a week shortening and improving the code each time). In addition, I investigated the mathematics and derived all formulae for the $K = 3$ case and confirmed that the proposed sample procedure does not give a uniform distribution on $\diamond^{K-1}$. To show the difference of the result of samples from each approach we show $K = 3$ uniform Dirichlet parameter samples for $L_k \geq 0.1$ and $U_k \leq 0.8$ for all $k = 1, 2, 3$ in Figure 6.18. The sample on the left is produced from the probability integral transform approach with x_2 used as the variate for the marginal. The yellow dots represent the sample and the black dot their mean. If the sample were uniform the black dot should be in the center of the equilateral triangle defined by the constraints. It can be seen to be off center toward x_2 and careful inspection of the sample shows that the density of the yellow points is higher in this corner. In contrast the sample on the right is from the proposed Gibbs sampler and it can be seen to be uniform.

Therefore, we implement the Gibbs sampler in our code in order to randomly initialize particle matrices on $\diamond^{K-1}$. We summarize the swarm initialization for PSO searches on $\diamond^{K-1}$ in Algorithm 6.6.

6.7.2.3 Particle confinement on $\diamond^{K-1}$ The concept of our algorithm to confine design points to $\diamond^{K-1}$ if they leave the polytope during the search is

1. Trace a line from the violating design point $\mathbf{x}' \notin \diamond^{K-1}$ to the centroid of the polytope denoted $\mathbf{c} \in \diamond^{K-1}$.
2. Map the particle to the point of intersection between this line and the face of

Algorithm 6.6: CprodSimplex-PSO on $\diamond^{K-1}$: random uniform swarm initialization of positions.

- 1: **Input:** $N, K, S, \mathbf{l}, \mathbf{u}$.
- 2: **Output:** $\{\mathbf{X}_i\}_{i=1}^S$: a set of S constrained random mixture designs on $\times_{j=1}^N \diamond^{K-1}$.
- 3: **for each** $i = 1, \dots, S$ **do**
- 4: $\mathbf{X}_i \sim \prod_{j=1}^N \text{TD}(\mathbf{x}' | \mathbf{1}_K; \mathbf{l}, \mathbf{u})$
- 5: $\mathbf{V}_i \sim \text{clampVelocityPolytope}(\prod_{j=1}^N \text{Dir}(\mathbf{x}' | \mathbf{1}_K))$
- 6: **endfor**
- 7: **return**

$\diamond^{K-1}$.

The algorithm commences by finding that dimension k which has the largest distance (absolute) from the boundaries L_k or U_k , maps that component to the boundary and adjusts the other components proportionally toward the centroid. We call this procedure `polytopeConfine` and present it in Algorithm 6.7.

6.8 Validating CprodSimplex-PSO on $\diamond^{K-1}$

Due to the complexity of this problem, there are limited references against which to run cases and demonstrate the efficiency and efficacy of CprodSimplex-PSO on $\diamond^{K-1}$. We list the references we found discussing generation of optimal designs (exact or continuous) below:

1. Wong et al. (2015) *A modified particle swarm optimization technique for finding optimal designs for mixture models* [94]. This paper shows the development of a PSO algorithm, which uses standard Euclidean geometry in the update equations, for generating continuous optimal designs for mixture experiments. While these authors addressed a different problem than exact design generation,

Algorithm 6.7: CprodSimplex-PSO on $\diamond^{K-1}$: confine particles to the boundary (edges and faces) defined by $\diamond^{K-1}$.

```

1: Input:  $\{\mathbf{X}_i\}_{i=1}^S$ ,  $\mathbf{c}$ ,  $\mathbf{l}$ ,  $\mathbf{u}$ .
2: Output:  $\{\mathbf{X}_i\}_{i=1}^S$ : all rows of candidate designs constrained to  $\times_{j=1}^N \diamond^{K-1}$ .
3: for each  $i = 1, \dots, S$  do
4:   for each  $j = 1, \dots, N$  do
5:     // check elementwise for violations of the polytope boundaries
6:      $v_L \leftarrow \text{any}(\mathbf{x}'_{ij} < \mathbf{l})$ ;  $v_U \leftarrow \text{any}(\mathbf{x}'_{ij} > \mathbf{u})$ 
7:     if  $!(v_L \vee v_U)$  // if no boundaries are violated return the point
8:       return  $\mathbf{x}'_{ij}$ 
9:     else // else adjust the point
10:      while  $(v_L \vee v_U)$  // run until particle is not on the face of  $\diamond^{K-1}$ 
11:        //compute ranges and particle's standardized distance from boundaries
12:         $\mathbf{r} = \mathbf{l} - \mathbf{u}$ ;  $\mathbf{l}_\Delta = (\mathbf{l} - \mathbf{x}'_{ij}) ./ \mathbf{r}$ ;  $\mathbf{u}_\Delta = (\mathbf{x}'_{ij} - \mathbf{u}) ./ \mathbf{r}$ 
13:        //record elementwise which max delta occurs
14:         $\mathbf{m}_\Delta \leftarrow \max.(\mathbf{l}_\Delta, \mathbf{u}_\Delta)$ 
15:        // record indicator of whether max distance dim to  $\diamond^{K-1}$  was to  $L$  or  $U$ 
16:         $\mathbf{u}.\text{or}.\mathbf{l}\{\text{which}.\text{ind}(\mathbf{m}_\Delta == \mathbf{l}_\Delta)\} \leftarrow -1$ 
17:         $\mathbf{u}.\text{or}.\mathbf{l}\{\text{which}.\text{ind}(\mathbf{m}_\Delta == \mathbf{u}_\Delta)\} \leftarrow 1$ 
18:        // record indices where  $\mathbf{m}_\Delta$  occurs, and does not occur
19:         $\text{ind}.\text{max} \leftarrow \text{which}.\text{max}(\mathbf{m}_\Delta)$ ;  $\text{ind}.\text{not}.\text{max} \leftarrow \{1, 2, \dots, K\} \setminus \text{ind}.\text{max}$ 
20:        if  $\mathbf{u}.\text{or}.\mathbf{l}\{\text{ind}.\text{max}\} == 1$ 
21:           $\beta = (\mathbf{u}\{\text{ind}.\text{max}\} - \mathbf{c}\{\text{ind}.\text{max}\}) / (\mathbf{x}'_{ij}\{\text{ind}.\text{max}\} - \mathbf{c}\{\text{ind}.\text{max}\})$ 
22:           $\mathbf{x}^{*'}_{ij}\{\text{ind}.\text{max}\} \leftarrow \mathbf{u}\{\text{ind}.\text{max}\}$ 
23:        elseif  $\mathbf{u}.\text{or}.\mathbf{l}\{\text{ind}.\text{max}\} == -1$ 
24:           $\beta = (\mathbf{l}\{\text{ind}.\text{max}\} - \mathbf{c}\{\text{ind}.\text{max}\}) / (\mathbf{x}'_{ij}\{\text{ind}.\text{max}\} - \mathbf{c}\{\text{ind}.\text{max}\})$ 
25:           $\mathbf{x}^{*'}_{ij}\{\text{ind}.\text{max}\} \leftarrow \mathbf{l}\{\text{ind}.\text{max}\}$ 
26:        endif
27:         $\mathbf{x}^{*'}_{ij}\{\text{ind}.\text{not}.\text{max}\} \leftarrow \beta(\mathbf{x}'_{ij}\{\text{ind}.\text{not}.\text{max}\}) + (1 - \beta)\mathbf{c}\{\text{ind}.\text{not}.\text{max}\}$ 
28:        //update while loop run condition
29:         $v_L \leftarrow \text{any}(\mathbf{x}^{*'}_{ij} < \mathbf{l})$ ;  $v_U \leftarrow \text{any}(\mathbf{x}^{*'}_{ij} > \mathbf{u})$ 
30:      endwhile
31:    endif
32:  endfor
33: endfor
34: return

```


there is one published D -optimal mixture design with mixture constraints where the support probabilities were uniform. Therefore we can attempt to duplicate these results for exact designs.

2. Coetzer and Haines (2017) *The construction of D and IV -optimal designs for mixture experiments with linear constraints on the components* [30]. This paper highlights that generating optimal designs for mixture experiments, especially on high dimensional polytopes, is a difficult problem. The authors claim a new approach that appeals to a representation of $\diamond^{K-1}$ as a convex combination of the vertices. They develop the theory which relies on barycentric coordinates describing the polytope, and they propose an algorithm based on this theory to generate optimal designs. They publish a few D and IV -optimal designs with constraints that we may compare our results to. These authors make brief mention to the efficiency of the algorithm: in a D -optimal with $N = 12$ and the full cubic Scheffé model search they ran the algorithm 1000 times and observe the optimal design in 515 runs. In the same D -optimal search but for $N = 15$ points they only found the optimal design in 42 of the 1000 runs.
3. Limmun, Borkowski, and Chomtee (2019) *Using a genetic algorithm to generate D -optimal designs for mixture experiments*. [61]. These authors focus on the generation of D -optimal designs with a Genetic Algorithm (GA) as the search algorithm. GA and PSO are both in the general family of evolutionary algorithms, however, they function very differently. These authors report that GA demonstrates superior efficacy to more mainstream algorithms, such as the extreme vertices design generator and exchange point algorithms, for generating D -optimal designs in $\diamond^{K-1}$. The authors publish a large set of GA optimal designs for a number of N and model scenarios against which we can validate

the proposed PSO algorithm.

6.8.1 Validation runs using the D -criterion

All PSO searches were run using SPSO 2007 (local communication topology), $S = 150$ particles and $n_{\text{run}} = 210$ searches parallelized over 7 computing cores.

The D -eff computed as in Limmun et al. (2012) is

$$D\text{-eff} = \frac{100|\mathbf{F}'\mathbf{F}|^{\frac{1}{p}}}{N}$$

and we note that on this scale a larger score is better. We have reported the search data in the four following examples on this scale.

6.8.1.1 $K = 3$, $N = 9$, cubic model without 3-way interaction This example comes from Wong et al. (2015) [94]. The only constraint in this design is $x_1 \leq 0.5$. Note that while this paper focuses solely on PSO generation of continuous designs for mixture experiments, this particular design scenario resulted in a design that has uniform support distribution and so we can use it to validate our algorithm for generating exact designs.

This set of runs took approximately 4 minutes to complete. We present the results of this search in Figure 6.19. The top left panel shows the distribution of the D -score for the 210 PSO runs, and shows that all 210 PSO runs found an equally good optimal design (there is no variation in these values). The top right panel shows the distribution of number of iterations and on median it only took CprodSimplex-PSO 430 iterations to find the design. The bottom left panel shows the design reported by Wong et al. while the bottom right panel shows the PSO generated design. These designs are the same.

6.8.1.2 $K = 3$, $N = 7$, second-order model This example comes from Limmun et al. (2012) [61]. The constraints stated for this design are

1. $0.3 \leq x_1 \leq 0.8$,
2. $0.0 \leq x_2 \leq 0.5$,
3. $0.0 \leq x_3 \leq 0.5$.

We present the results of the searches in Figure 6.20. We can see that most but not all PSO searches found the same design (there are 2 solutions shown in the boxplot of D -eff top-left panel). Right top panel shows that these searches took on median 720 iterations. Bottom right panel shows the optimal design produced by Limmun et al. (2012) while the bottom right panel shows the PSO generated design. These designs are the same.

This set of runs took approximately 5 minutes to complete.

6.8.1.3 $K = 4$, $N = 8$, first-order model This example comes from Limmun et al. (2012) [61]. The constraints stated for this design are

1. $0.4 \leq x_1 \leq 0.6$,
2. $0.1 \leq x_2 \leq 0.47$,
3. $0.1 \leq x_3 \leq 0.47$,
4. $0.03 \leq x_4 \leq 0.08$.

We present the results of the searches in Figure 6.21. We can see that most but not all PSO searches found the same design (there is now a fair amount of variation in the boxplot of D -eff top-left panel). Right top panel shows that these searches took on median 630 iterations. Bottom right panel shows the optimal design produced by

Limmun et al. (2012) while the bottom right panel shows the PSO generated design (it can be seen that these are the same design).

This set of runs took approximately 6 minutes to complete.

6.8.1.4 $K = 4$, $N = 21$, first-order model This example comes from Limmun et al. (2012) [61]. The constraints stated for this design are

1. $0.4 \leq x_1 \leq 0.6$,
2. $0.1 \leq x_2 \leq 0.47$,
3. $0.1 \leq x_3 \leq 0.47$,
4. $0.03 \leq x_4 \leq 0.08$.

We present the results of the searches in Figure 6.22. We can see that most but not all PSO searches found the same design (there is now a fair amount of variation in the boxplot of D -eff top-left panel). Right top panel shows that these searches took on median 1000 iterations.

The design reported by Limmun et al. (2012) has D -eff = 0.015728 while the best PSO design has D -eff = 0.015830 which is slightly better. We present the generated design in Table 6.5.

This set of runs took approximately 22 minutes to complete.

6.8.2 Validation runs using the IV -criterion

Computing the IV -criterion for a candidate design involves significant complexity when working on $\diamond^{K-1}$. Some discussion of this can be found in Coetzer and Haines (2017) [30]. We need to compute both the volume

$$V(\diamond^{K-1}) = \int_{\diamond^{K-1}} \mathbf{d}\mathbf{x} \quad (6.54)$$

and the design moment matrix

$$\mathbf{W} = \int_{\Diamond^{K-1}} \mathbf{f}(\mathbf{x}') \mathbf{f}'(\mathbf{x}') d\mathbf{x}'. \quad (6.55)$$

Determining the bounds of integration over $\Diamond^{K-1}$ is, in general, challenging. Therefore, for a fixed design scenario, we implemented a straight-forward Monte-Carlo integration. This is possible because V and $\mathbf{W}$ are properties of the model and the space $\Diamond^{K-1}$ and can be computed before implementing the PSO search and are used inside the objective function to evaluate all candidate designs for a fixed K, N , and constraints design scenario.

Coetzer and Haines (2017) [30] provide only one example against which we can validate our approach. The problem is a 4-D IV-optimal design problem with constraints

1. $0.2 \leq x_1 \leq 0.65$,
2. $0.1 \leq x_2 \leq 0.55$,
3. $0.1 \leq x_3 \leq 0.2$,
4. $0.15 \leq x_4 \leq 0.35$.

and they produced designs for $N = 10, 15$ scenarios.

6.8.2.1 $N = 10$ We ran the CprodSimplex-PSO algorithm on this problem and were able to reproduce (to 3 decimal places) the currently known IV-optimal design. We present the performance statistics in Figure 6.23. We present a direct comparison of the Coetzer and Haines (2017) design versus the CprodSimplex-PSO generated design in Table 6.6.

To three decimal places CprodSimplex-PSO was able to produce the best known IV -optimal design and was able to do it on median at 1000 iterations.

It took approximately 6 minutes to complete this set of PSO searches.

6.8.2.2 Second order model, $K = 4$, $N = 15$ runs. We ran the CprodSimplex-PSO algorithm on this problem and were close to reproducing the currently known IV -optimal design.

For the CprodSimplex-PSO we attained an optimal design with average prediction variance $IV\text{-PSO} = 0.3532$ while the design published by Coetzer and Haines (2017) has a score $IV\text{-CH} = 0.3450$, and thus PSO design has 98% relative efficiency.

We present the performance statistics in Figure 6.24. We present a direct comparison of the Coetzer and Haines (2017) design versus the CprodSimplex-PSO generated design in Table 6.7.

CprodSimplex-PSO was able to produce this design on median at 1300 iterations.

It took approximately 21 minutes to complete this set of PSO searches.

6.9 Discussion and Conclusions

In this chapter we developed a novel PSO algorithm for optimizing functions that take matrix inputs over the Cartesian product of standard $K - 1$ -simplices. The algorithm relies foundationally on the Aitchison geometry which is a non-Euclidean geometry that induces a vector and metric-space on Δ^{K-1} . This was advantageous to developing a PSO algorithm because it works well with the definition of vectors in the space summing to 1 (e.g., the elements we are working with in mixture experiments are essentially discrete probability distributions).

The mathematics of the new algorithm, as well as all the sub-routines required to accomplish building this new algorithm, are fully described. This includes the

problem where each vector component is constrained to lower and upper bounds (creating the need to optimize functions on the Cartesian product of high-dimensional convex irregular polytopes).

We validated the algorithm against recent published results from the statistics literature for generating D and IV -optimal designs for mixture experiments. The results are unusually and unexpectedly good. We showed that, for the unconstrained simplex searches we could reproduce the best known designs, and the new PSO algorithm found this design *every time it was run*. Moreover, in part thanks to the computing speed of Julia as a compiled language, as well as parallel computing, the algorithm is quite fast and can be run a few hundred times on this problems in just a few minutes.

The algorithm also performed very well for generating optimal mixture designs where the i th component is constrained to upper and lower bounds L_i and U_i . For all cases where we searched for D -optimal designs, the algorithm found the solution (which matched or exceeded published results) quickly. We found just one published IV -optimal design in the literature against which we could test our proposed algorithm. We ran two cases (on the same design space) to generated IV -optimal designs of different size $N = 10, 15$. For the $N = 10$ case the algorithm found the published optimal design. For the $N = 15$ case the algorithm generated a design with 98% efficiency, which is highly successful. Regarding the last case: there are many input parameters to the algorithm that may be impacting the inability to find exactly the known optimal design. Further investigation is warranted in an effort to improve the decimal precision of the solution $\mathbf{X}$, and we intend to consider the following: the particle step size parameter, the relative tolerance stopping criteria which is currently implemented on f , varying the particle size S , and possible a hybrid approach where we take our solution, which is hopefully in the neighborhood

of the global optimizer, and implement a gradient search with our PSO-generated design as the starting point.

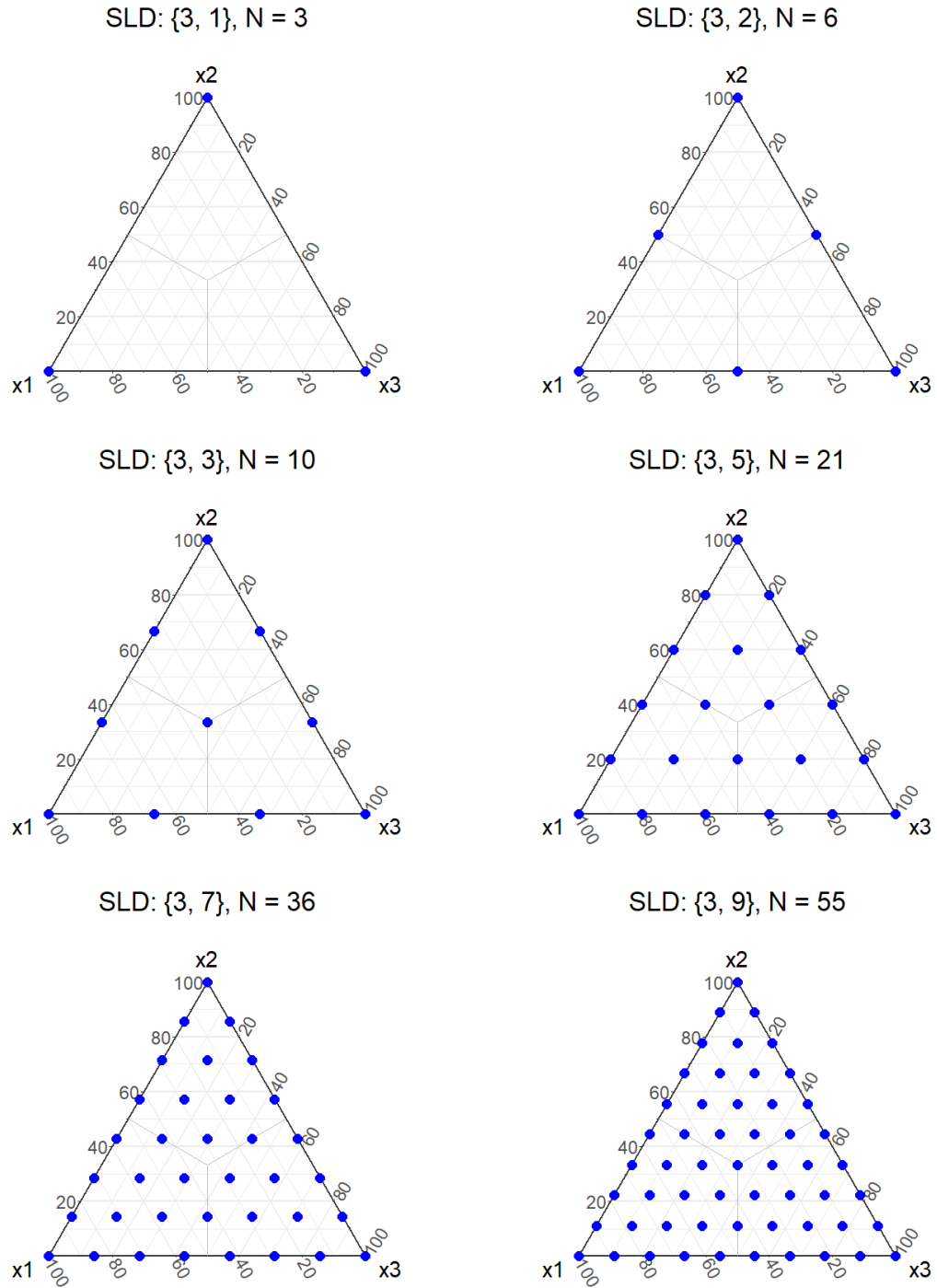


Figure 6.1: Simplex lattice designs for $K = 3$ and $m = 1, 2, 3, 5, 7, 9$.

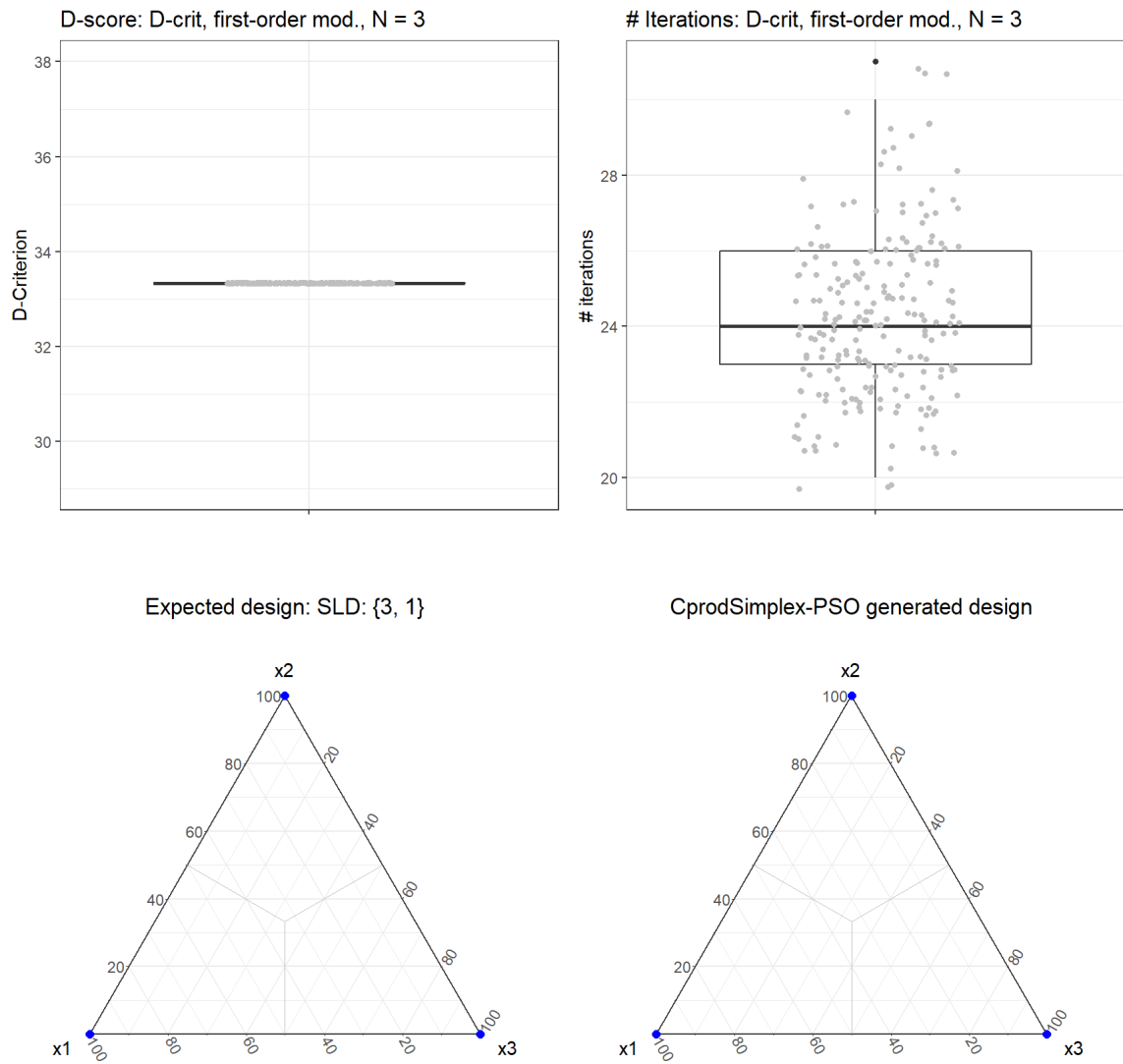


Figure 6.2: Results of the CprodSimplex-PSO search: $K = 3$, $N = 3$, first-order model.

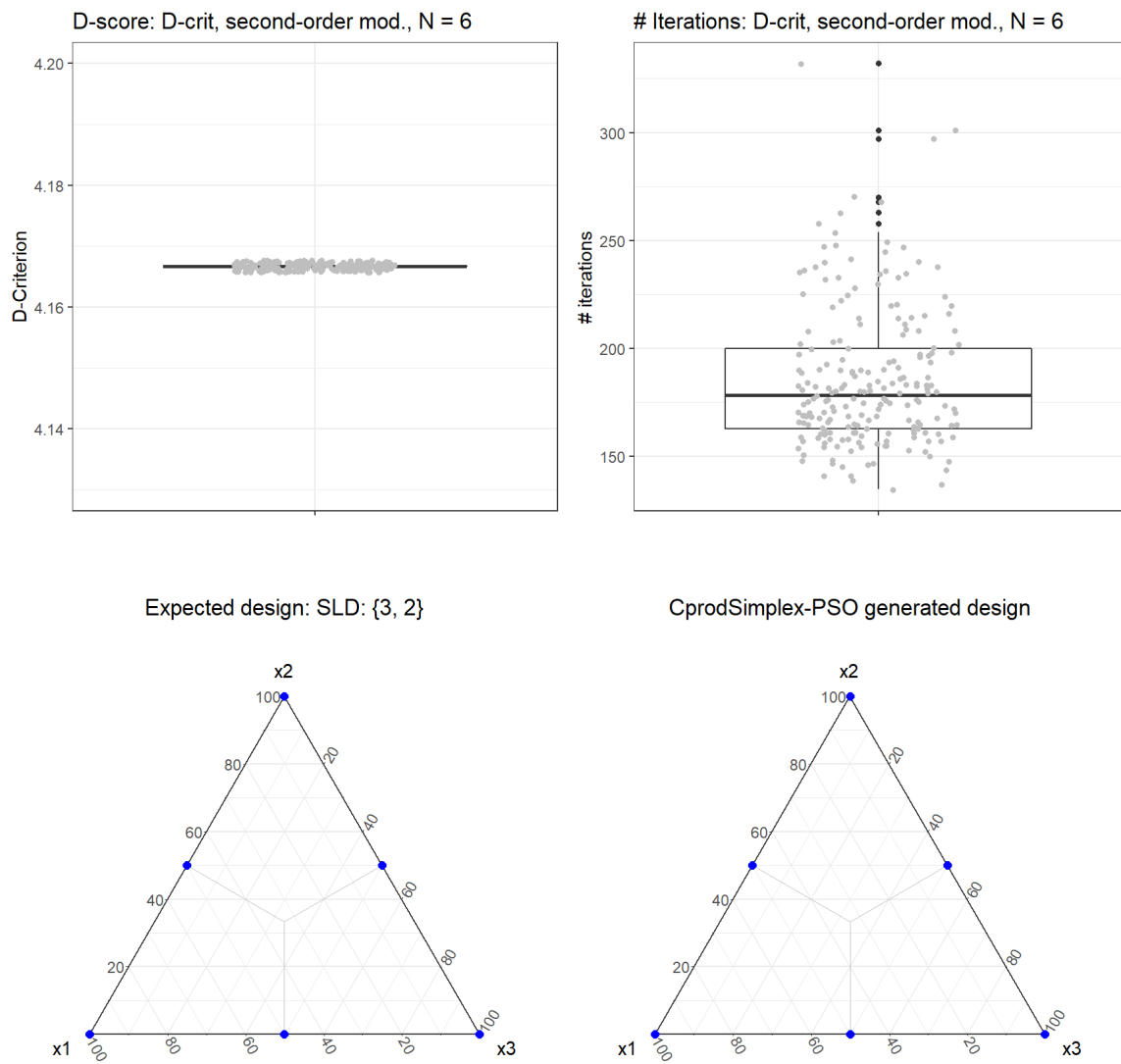


Figure 6.3: Results of the CprodSimplex-PSO search: $K = 3$, $N = 6$, second-order model.

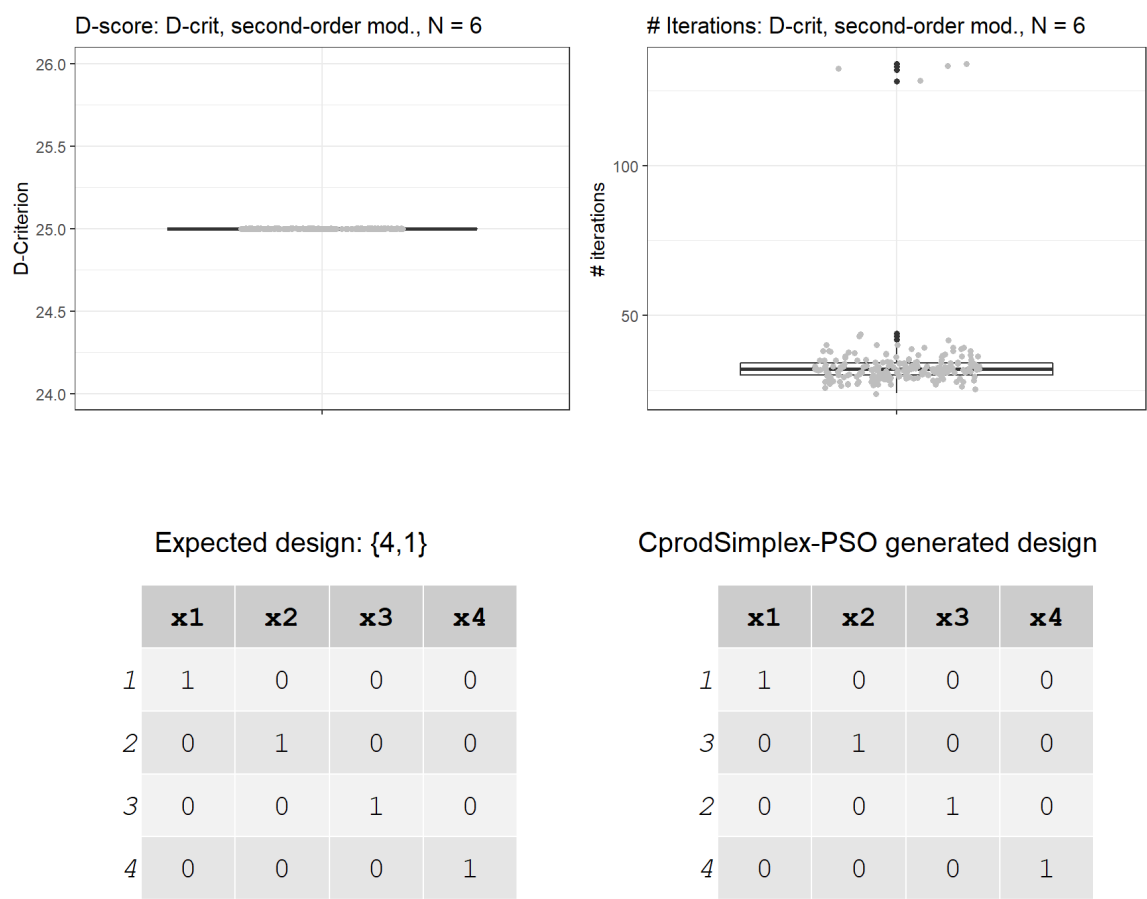


Figure 6.4: Results of the CprodSimplex-PSO search: $K = 4$, $N = 4$, second-order model.

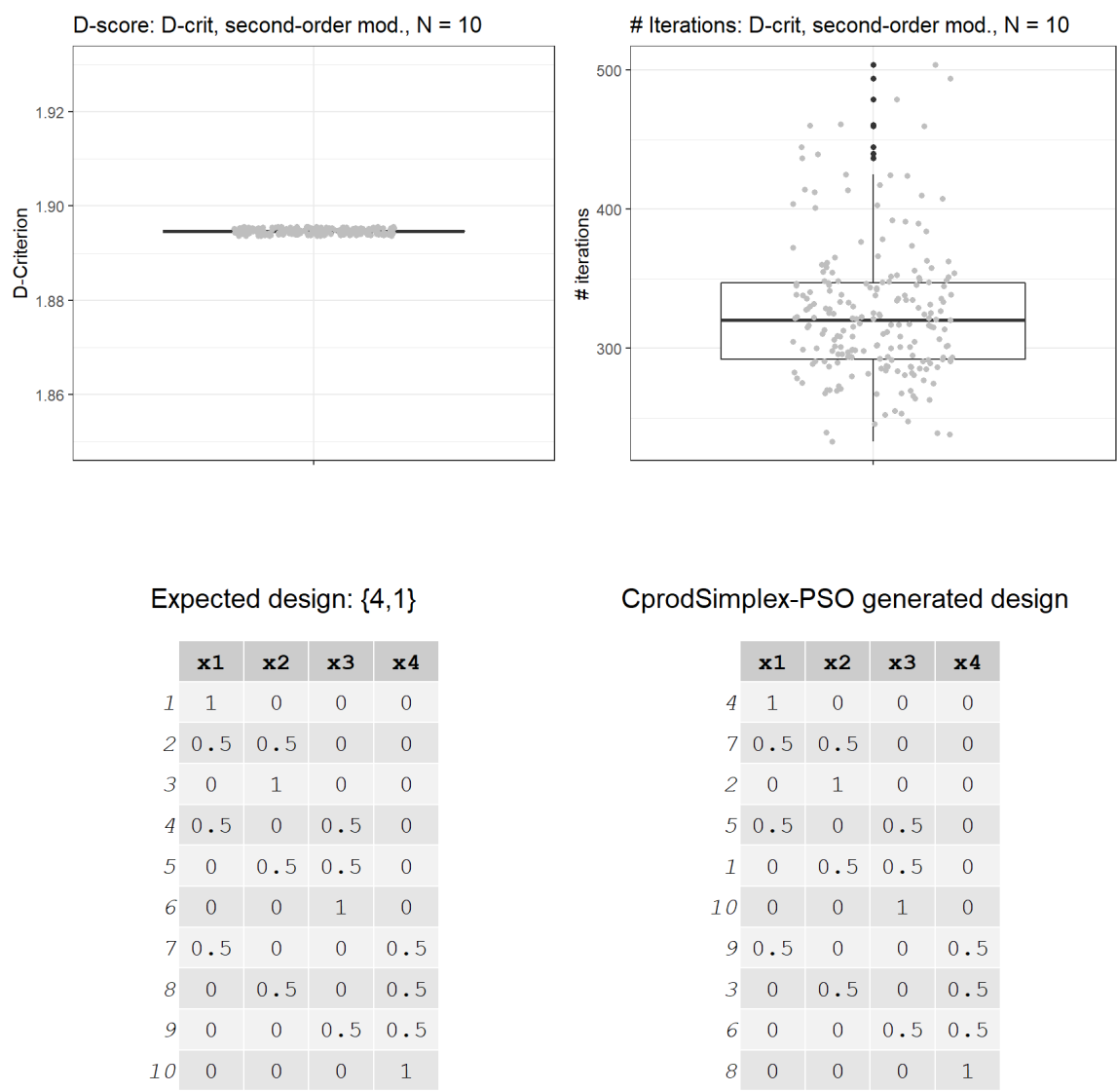


Figure 6.5: Results of the CprodSimplex-PSO search: $K = 4$, $N = 10$, second-order model.

SCD: $K = 3$, $N = 7$

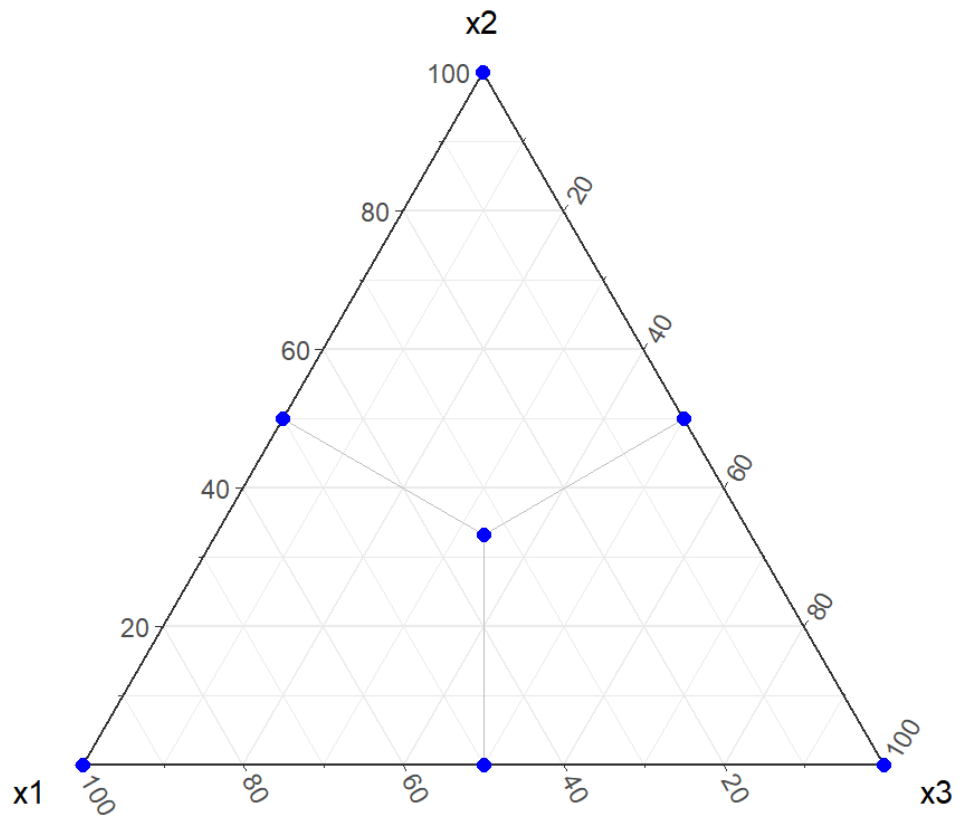


Figure 6.6: Simplex-centroid design for $K = 3$ factors.

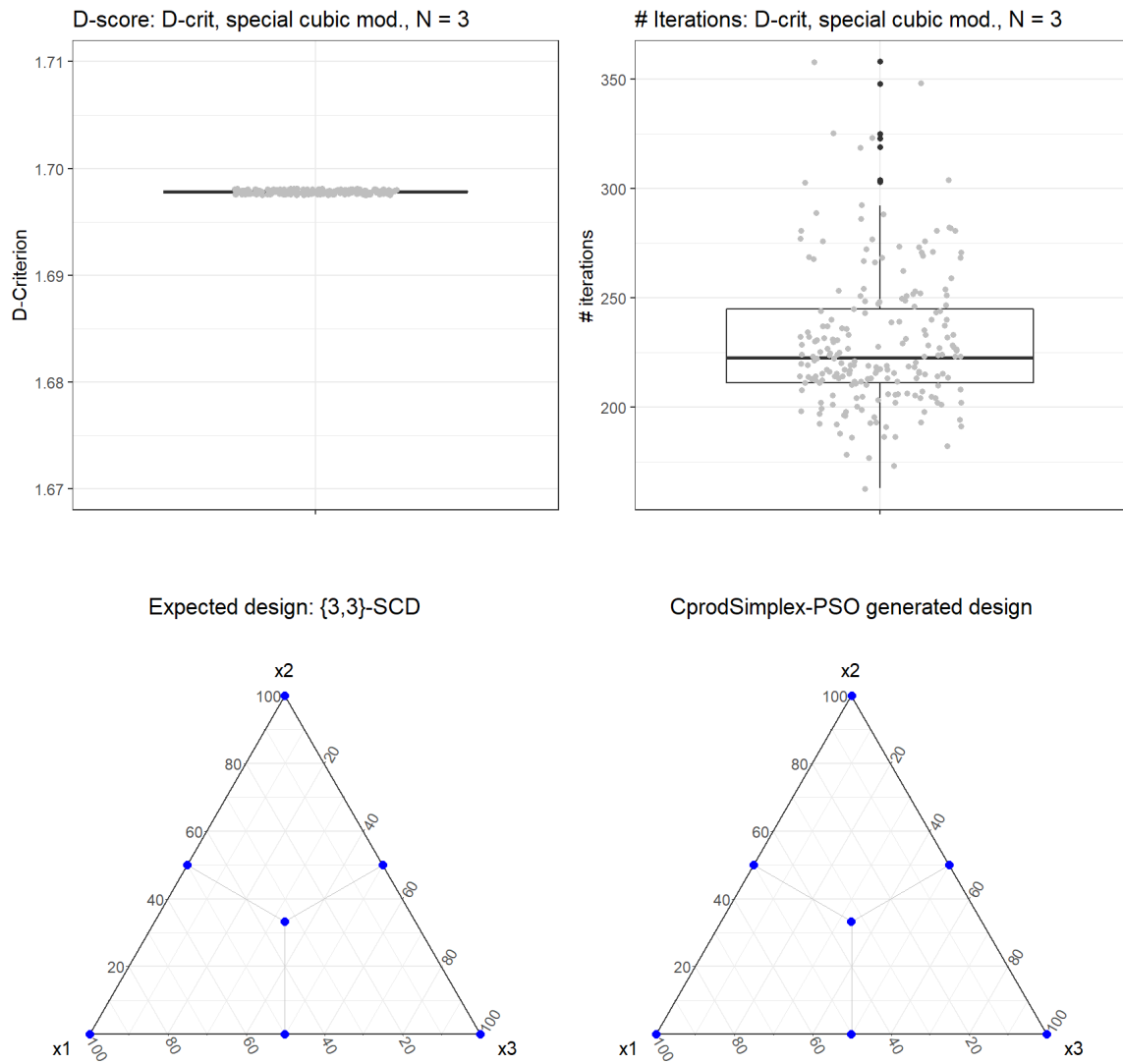


Figure 6.7: Results of the CprodSimplex-PSO search: $K = 3$, $N = 7$, special cubic model.

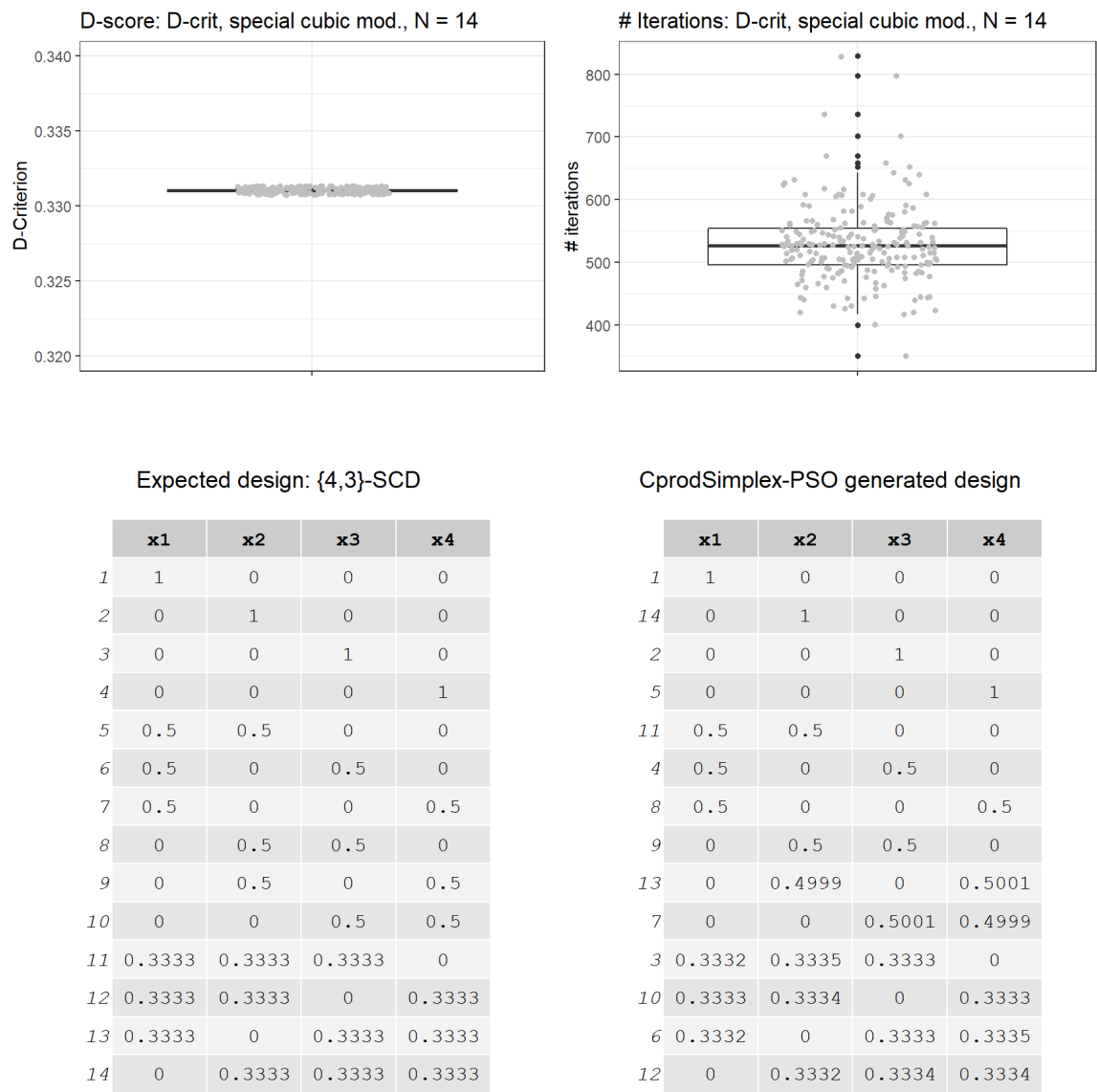


Figure 6.8: Results of the CprodSimplex-PSO search: $K = 4$, $N = 14$, special cubic model.

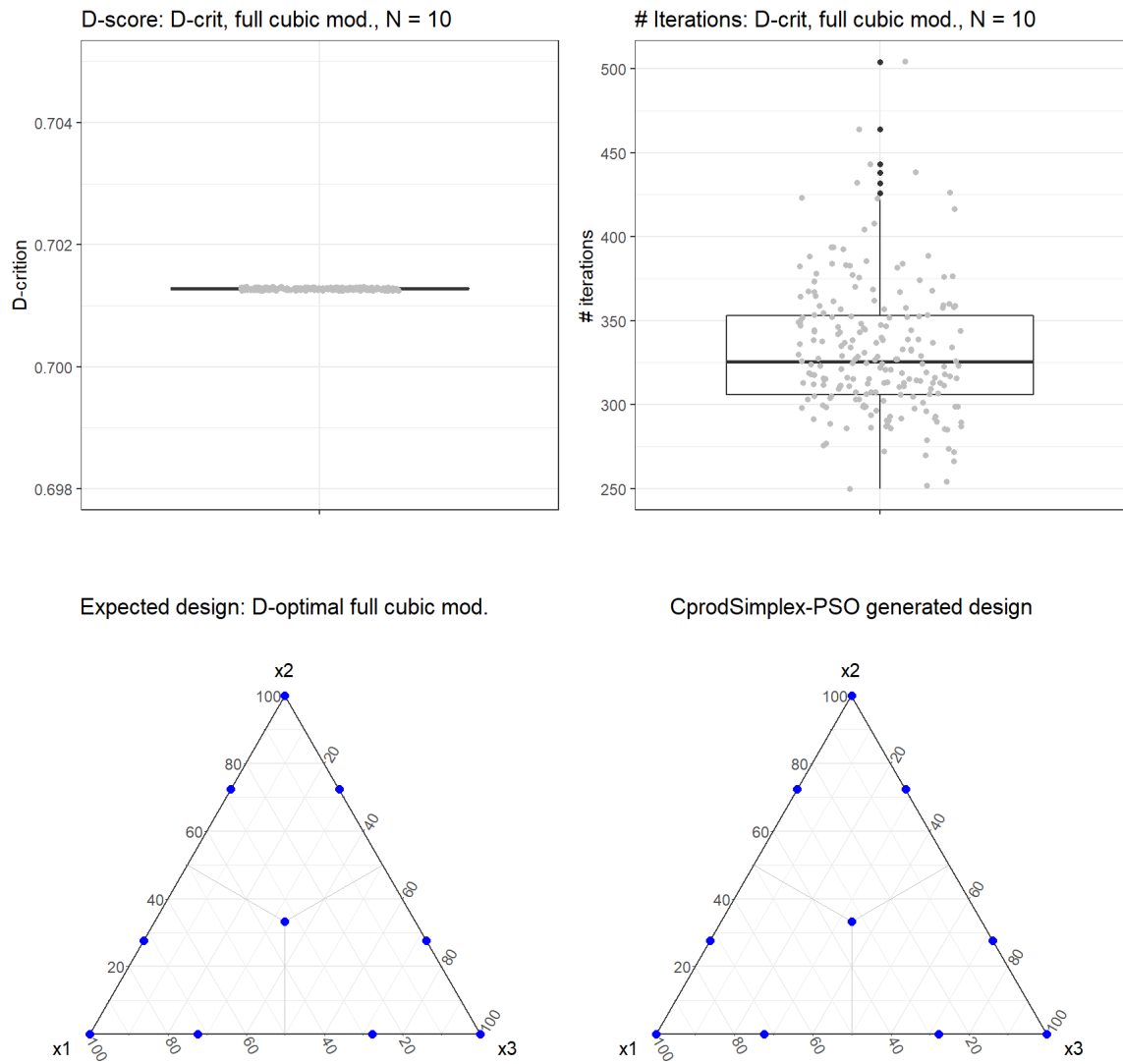


Figure 6.9: Results of the CprodSimplex-PSO search: $K = 3$, $N = 10$, full cubic model.

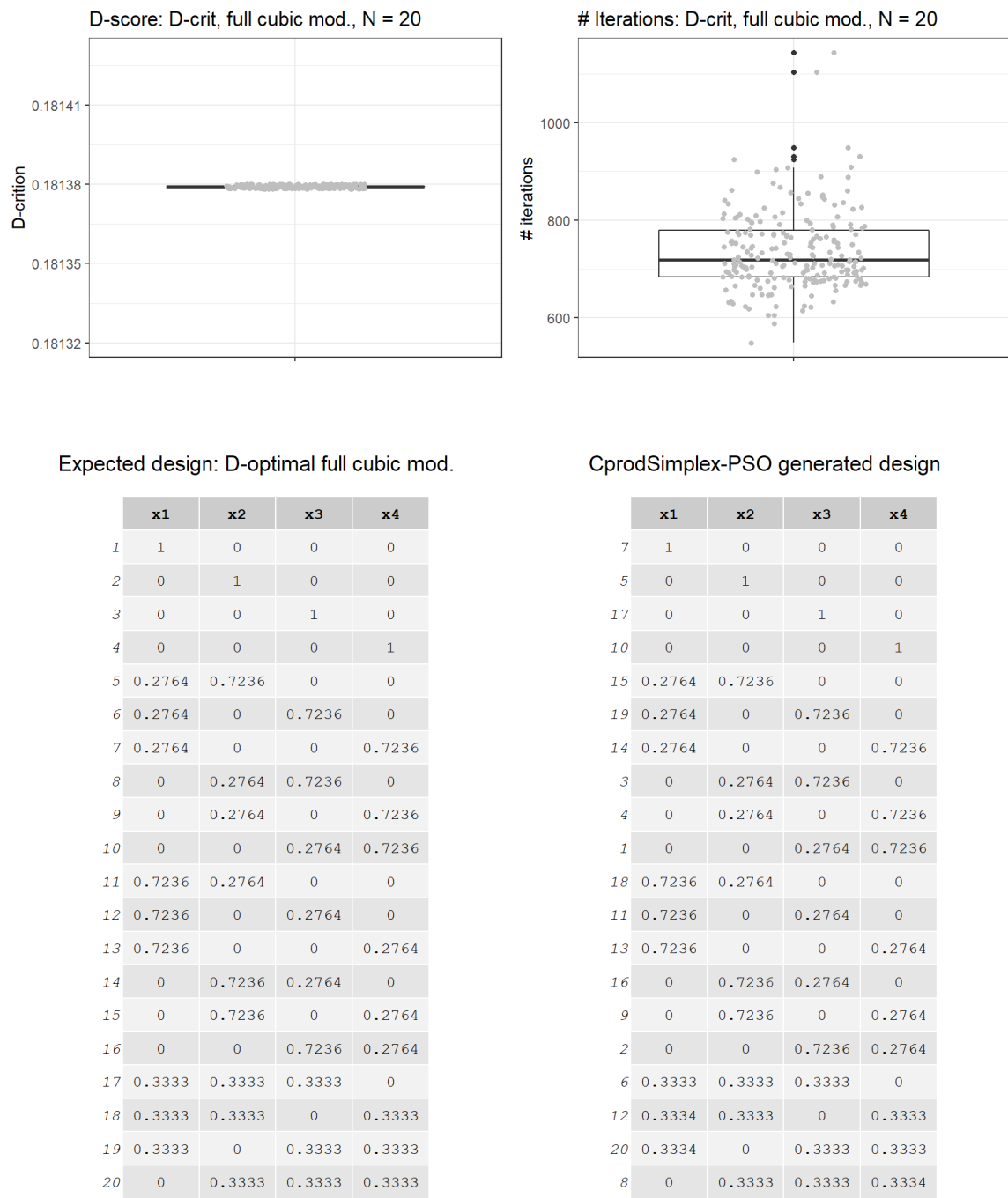


Figure 6.10: Results of the CprodSimplex-PSO search: $K = 4$, $N = 20$, full cubic model.

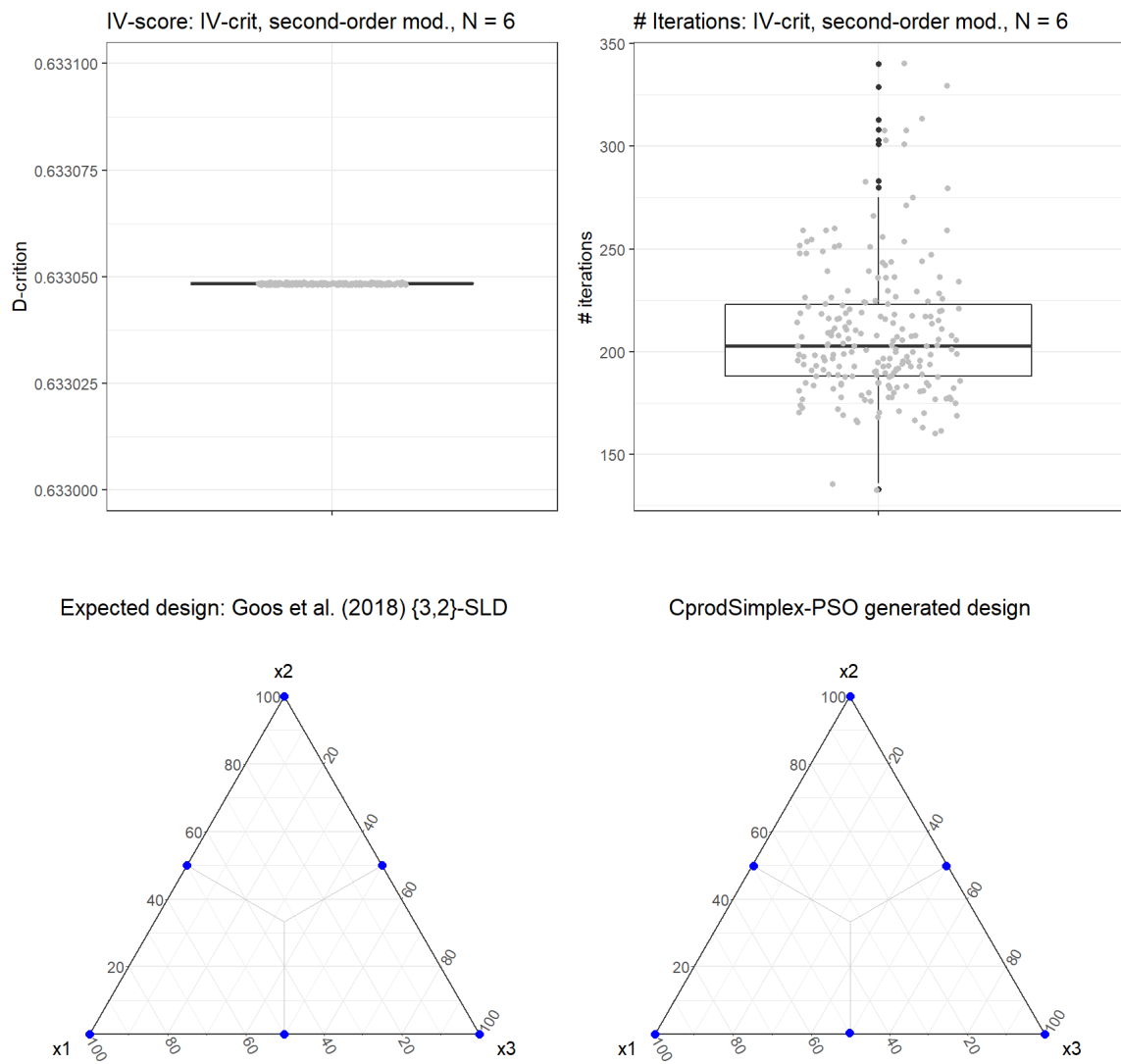


Figure 6.11: Results of the CprodSimplex-PSO search: $K = 3$, $N = 6$, second order model.

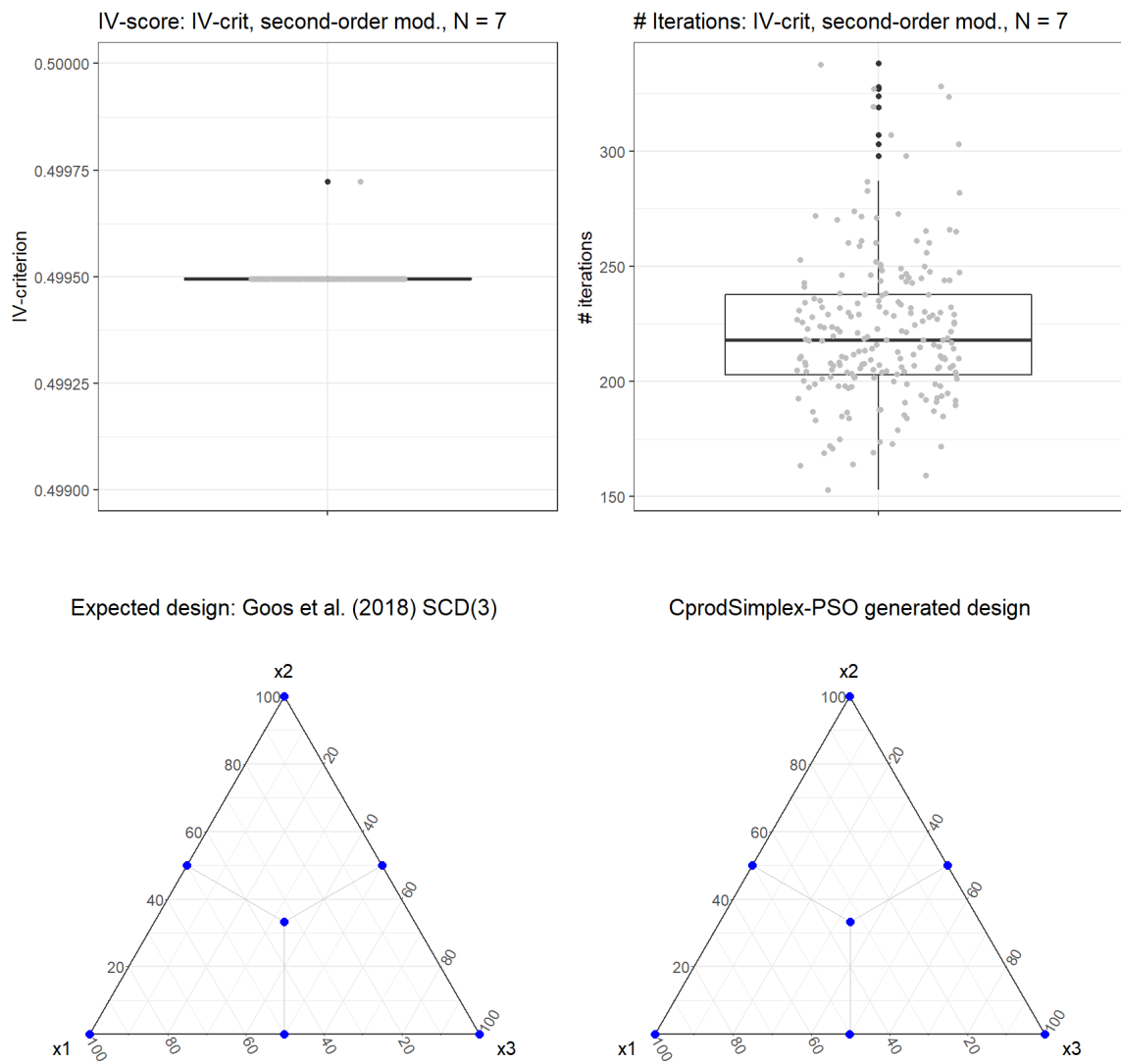


Figure 6.12: Results of the CprodSimplex-PSO search: $K = 3$, $N = 7$, second order model.

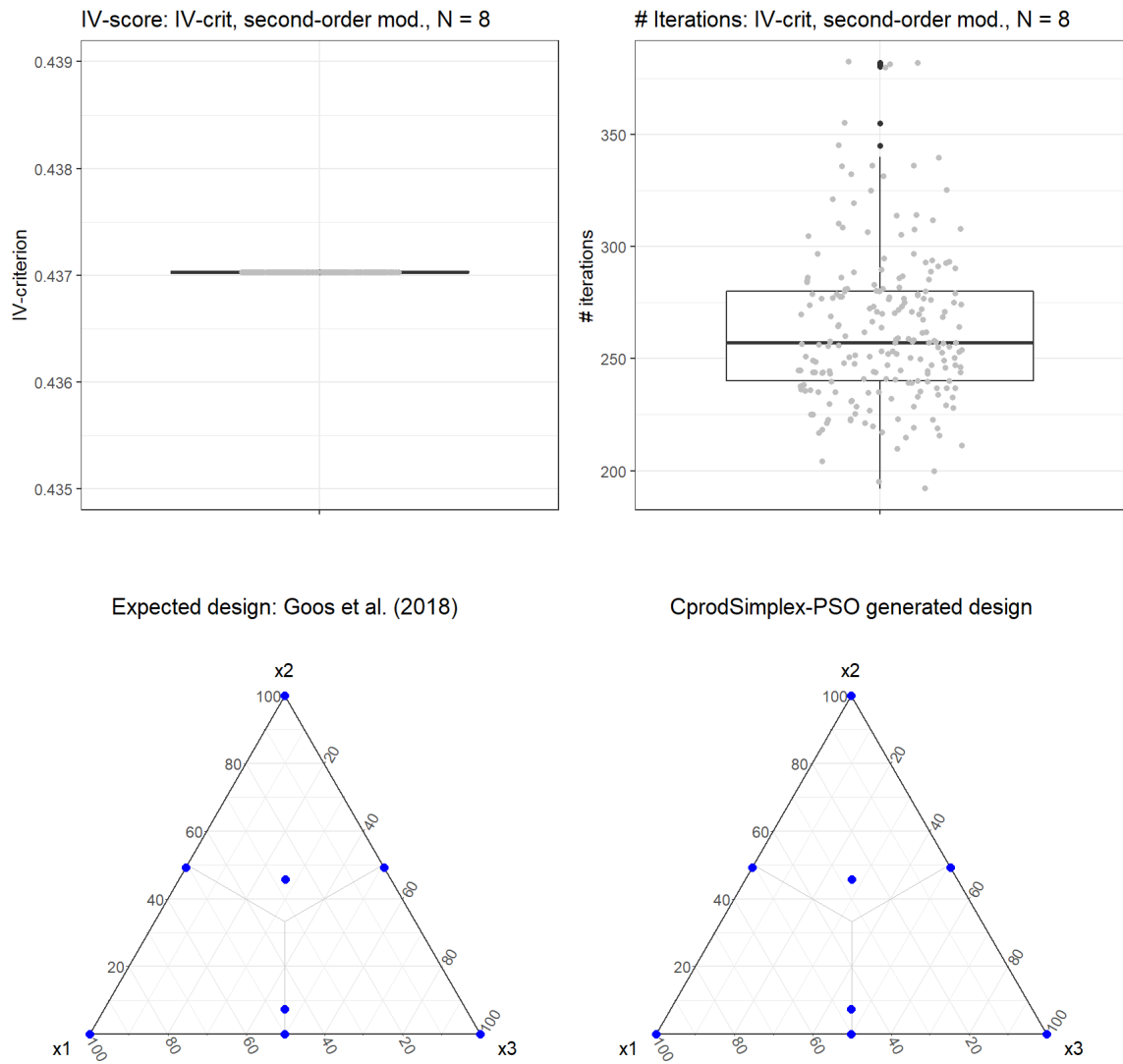
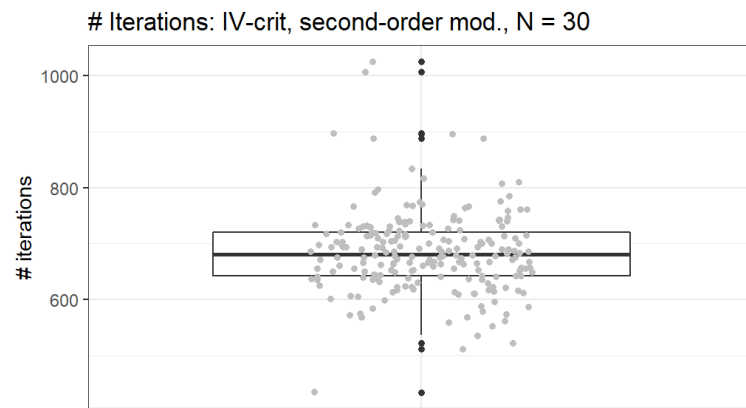
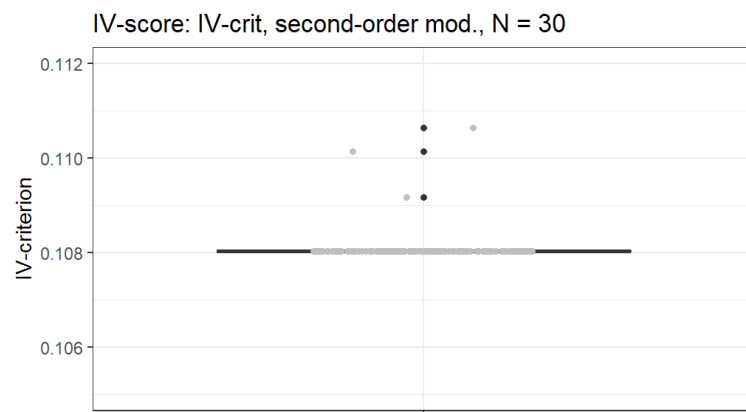


Figure 6.13: Results of the CprodSimplex-PSO search: $K = 3$, $N = 8$, second order model.



CprodSimplex-PSO generated design

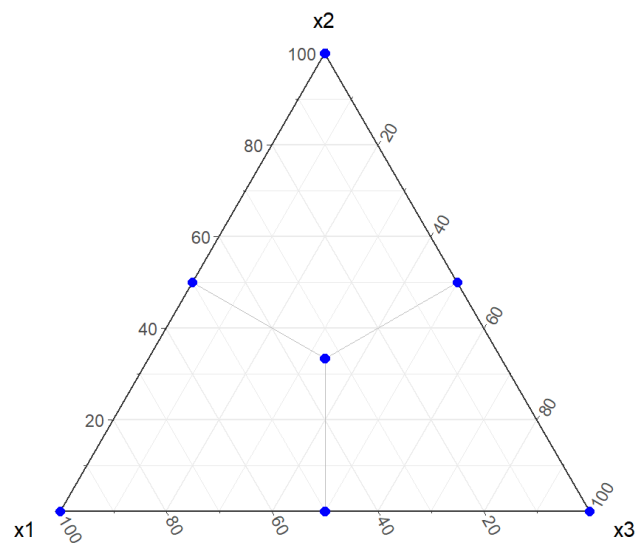


Figure 6.14: Results of the CprodSimplex-PSO search: $K = 3$, $N = 30$, second order model.

Table 6.1: The CprodSimplex-PSO generated IV -optimal design for $K = 3$, $N = 30$, second-order Scheffé model. N_i is the number of replicates of design point i .

IV -score = 0.11229				
component type	N_i	x_1	x_2	x_3
pure	3	1	0	0
pure	3	0	1	0
pure	3	0	0	1
ternary	3	0.3333	0.3333	0.3333
binary	6	0.5	0.5	0
binary	6	0.5	0	0.5
binary	6	0	0.5	0.5

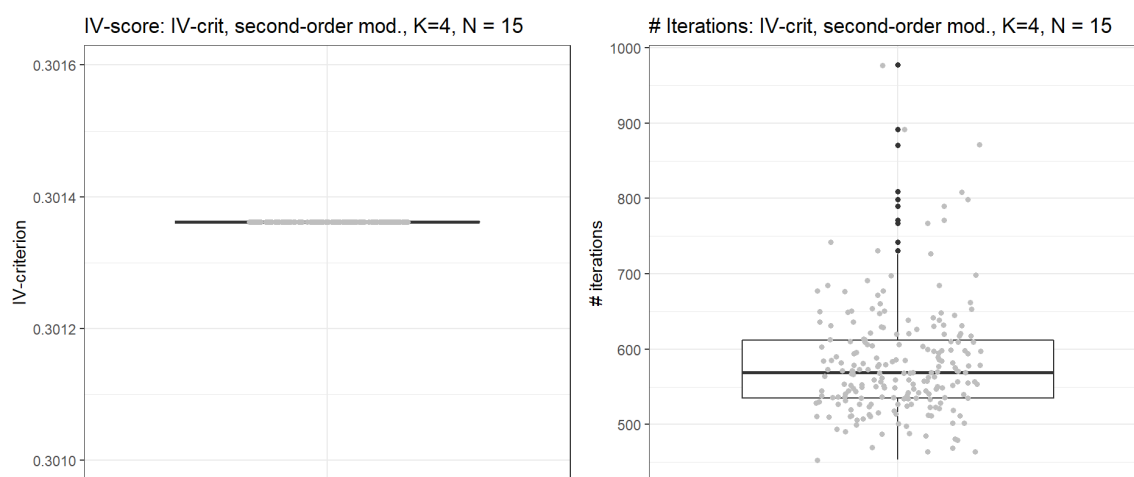


Figure 6.15: Results of the CprodSimplex-PSO search: $K = 4$, $N = 15$, second order model.

Table 6.2: The CprodSimplex-PSO generated *IV*-optimal design for $K = 4$, $N = 15$, second-order Scheffé model. The design points from the coordinate exchange algorithm (CEXCH) of Goos et al. (2016) are in parenthesis. The CprodSimplex-PSO design points are not in parenthesis. Discrepancy between the two optimal designs are denoted with an asterisk.

<i>IV</i> -CEXCH = 0.3022, <i>IV</i> -PSO = 0.301361					
index	x_1	x_2	x_3	x_4	discrepancy
1	1 (1)	0 (0)	0 (0)	0 (0)	
2	0 (0)	1 (1)	0 (0)	0 (0)	
3	0 (0)	0 (0)	1 (1)	0 (0)	
4	0 (0)	0 (0)	0 (0)	1 (1)	
5	0.5 (0.5)	0.5 (0.5)	0 (0)	0 (0)	
6	0.5 (0.5)	0.5 (0.5)	0 (0)	0 (0)	
7	0.5058 (0.5)	0 (0)	0 (0)	0.4942 (0.5)	*
8	0 (0)	0.5058 (0.5)	0.4942 (0.5)	0 (0)	*
9	0 (0)	0.5058 (0.5066)	0 (0)	0.4942 (0.4934)	*
10	0.5058 (0.505)	0 (0)	0.4942 (0.495)	0 (0)	*
11	0 (0)	0 (0)	0.5 (0.5)	0.5 (0.5)	
12	0.2876 (0.2876)	0.2876 (0.2876)	0.4247 (0.4248)	0 (0)	
13	0.2876 (0.2885)	0.2876 (0.2885)	0 (0)	0.4247 (0.423)	*
14	0.3321 (0.3331)	0 (0)	0.3339 (0.3335)	0.334 (0.3335)	*
15	0 (0)	0.3321 (0.3326)	0.334 (0.3337)	0.3339 (0.3337)	*

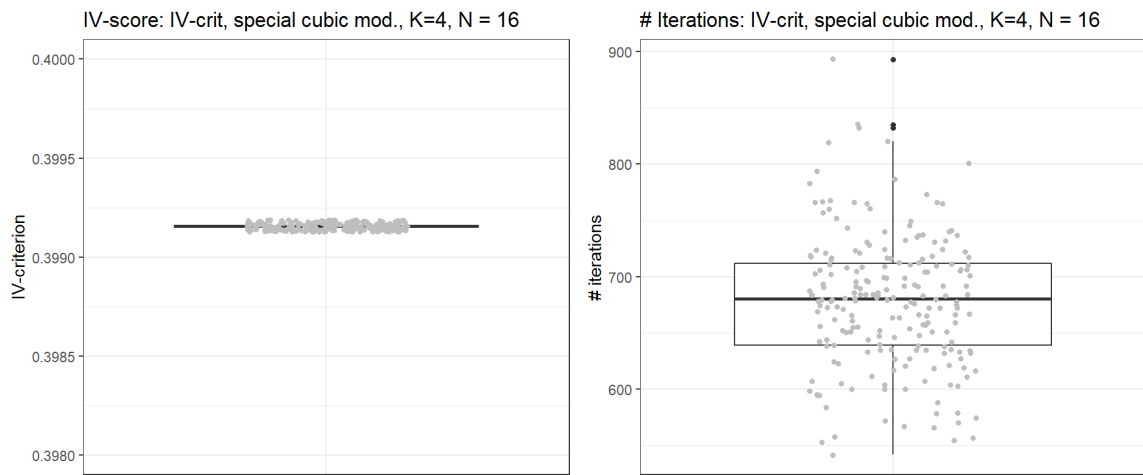


Figure 6.16: Results of the CprodSimplex-PSO search: $K = 4$, $N = 16$, special cubic model.

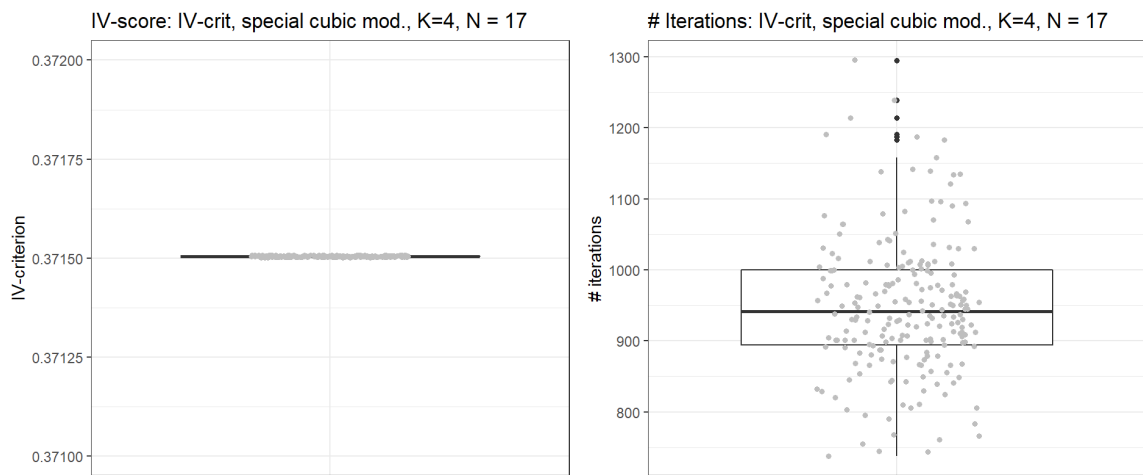


Figure 6.17: Results of the CprodSimplex-PSO search: $K = 4$, $N = 17$, special cubic model.

Table 6.3: The CprodSimplex-PSO generated IV -optimal design for $K = 4$, $N = 16$, special cubic model. The design matches that from the coordinate exchange algorithm (CEXCH) of Goos et al. (2016).

IV -PSO = 0.399157				
index	x_1	x_2	x_3	x_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0.5	0.5	0	0
6	0.5	0	0.5	0
7	0.5	0	0	0.5
8	0	0.5	0.5	0
9	0	0.5	0	0.5
10	0	0	0.5	0.5
11	0.3333	0.3333	0.3333	0
12	0.3333	0.3333	0	0.3333
13	0.3333	0	0.3333	0.3333
14	0	0.3333	0.3333	0.3333
15	0.25	0.25	0.25	0.25
16	0.25	0.25	0.25	0.25

Table 6.4: The CprodSimplex-PSO generated *IV*-optimal design for $K = 4$, $N = 17$, special cubic model. The design points from the coordinate exchange algorithm (CEXCH) of Goos et al. (2016) are in parenthesis. The CprodSimplex-PSO design points are not in parenthesis. Discrepancy between the two optimal designs are denoted with an asterisk.

IV -CEXCH = 0.371498, IV -PSO = 0.371504					
index	x_1	x_2	x_3	x_4	discrepancy
1	1 (1)	0 (0)	0 (0)	0 (0)	
2	0 (0)	1 (1)	0 (0)	0 (0)	
3	0 (0)	0 (0)	1 (1)	0 (0)	
4	0 (0)	0 (0)	0 (0)	1 (1)	
5	0.4987 (0.5)	0.5013 (0.5)	0 (0)	0 (0)	*
6	0.4998 (0.5)	0 (0)	0.5002 (0.5)	0.0000 (0)	*
7	0.4998 (0.5)	0 (0)	0 (0)	0.5002 (0.5)	*
8	0 (0)	0.5010 (0.5)	0.4990 (0.5)	0 (0)	*
9	0 (0)	0.5010 0	0 (0)	0.4990 (0.5)	*
10	0 (0)	0 (0)	0.5 (0.5)	0.5 (0.5)	
11	0.3344 (0.3358)	0.3288 (0.3284)	0.3368 (0.3358)	0.0000 (0)	*
12	0.3344 (0.3351)	0.3288 (0.3299)	0.0000 (0)	0.3368 (0.3351)	*
13	0.3336 (0.3338)	0.0000 (0)	0.3332 (0.3331)	0.3332 (0.3331)	*
14	0.3336 (0.3334)	0.0000 (0)	0.3332 (0.3333)	0.3332 (0.3333)	*
15	0.0000 (0)	0.3288 (0.3298)	0.3356 (0.3351)	0.3356 (0.3351)	*
16	0.2901 (0.2897)	0.3016 (0.3021)	0.2041 (0.2041)	0.2041 (0.2041)	*
17	0.1648 (0.1653)	0.3032 (0.3034)	0.2660 (0.2656)	0.2660 (0.2656)	*

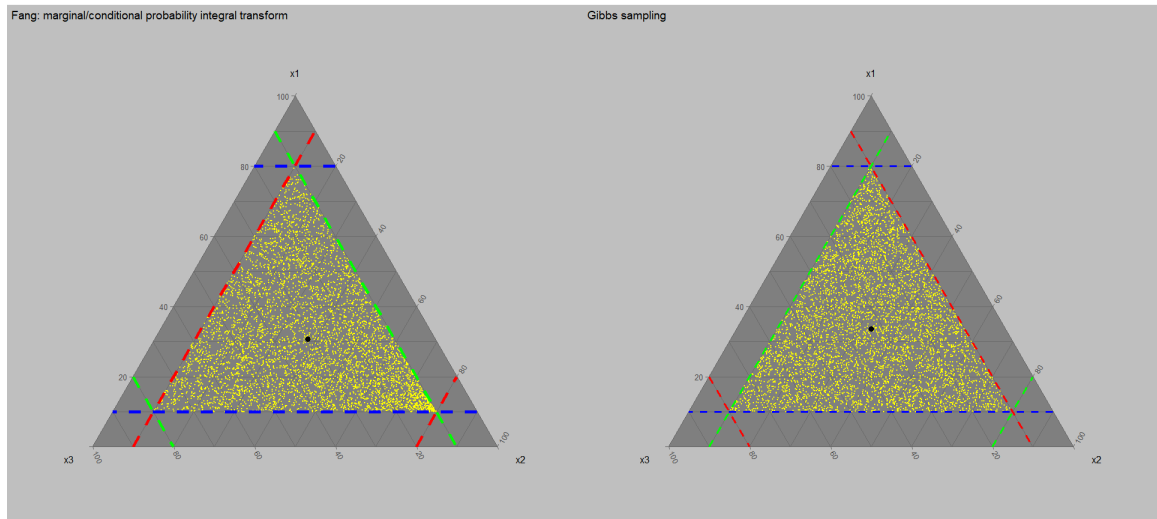


Figure 6.18: Left panel shows a sample using uniform Dirichlet parameters and the probability integral transform approach. The right hand panel shows a sample using uniform Dirichlet parameters and the Gibbs sampling approach.

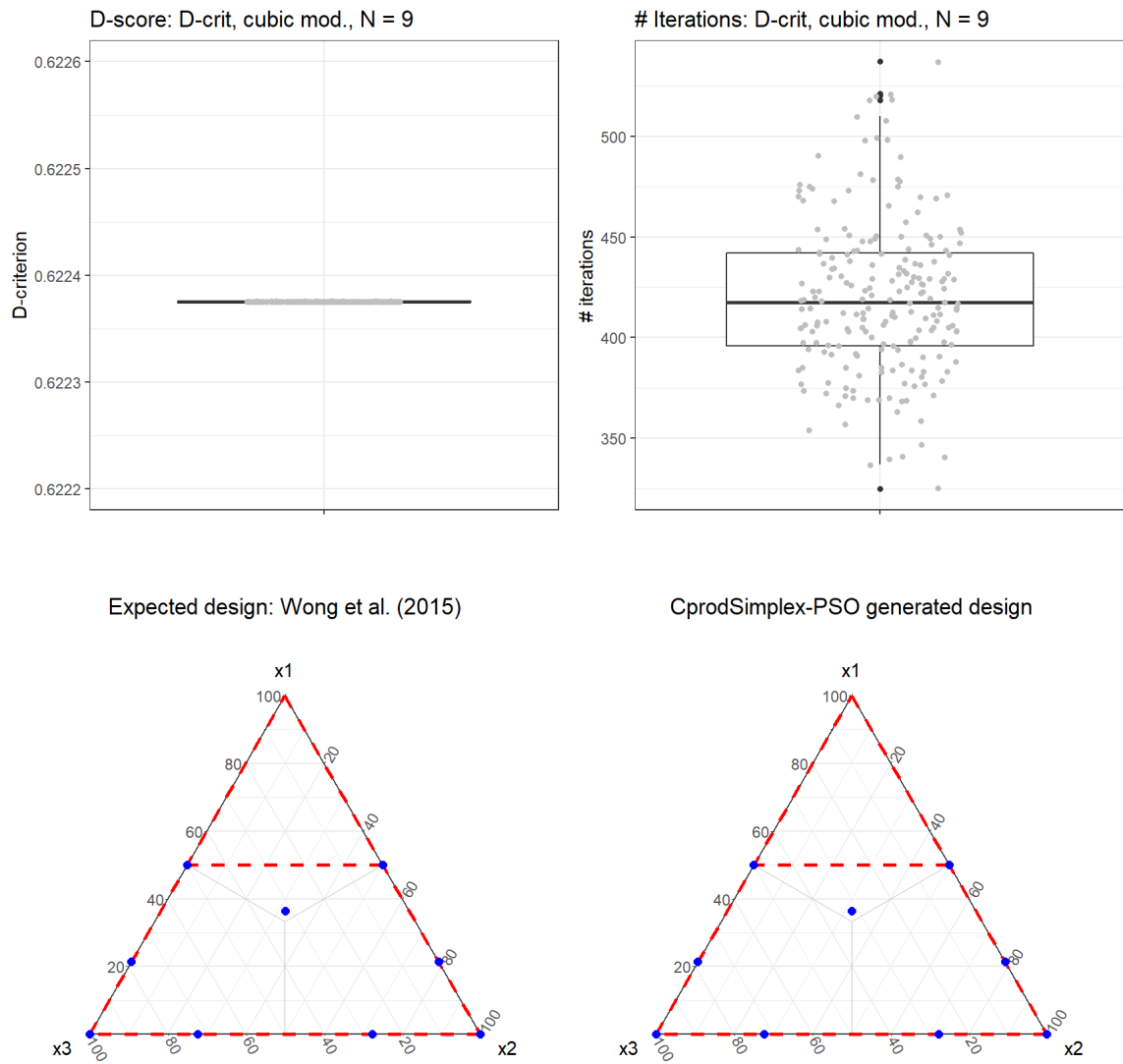


Figure 6.19: Results of the CprodSimplex-PSO search: $K = 3$, $N = 9$, modified cubic model (no 3-way interaction) and constraint $x_1 \leq 0.5$.

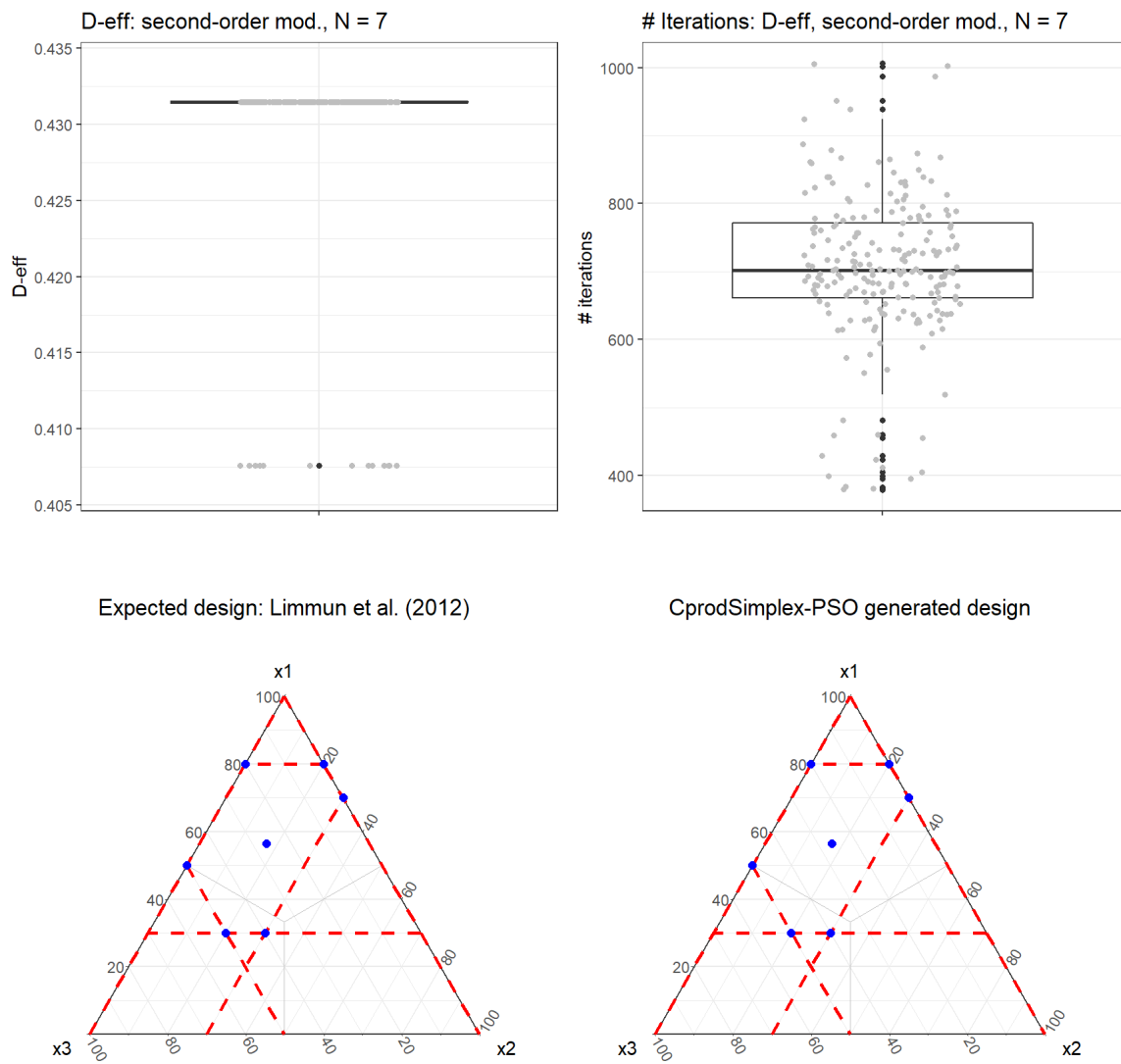


Figure 6.20: Results of the CprodSimplex-PSO search: $K = 3$, $N = 7$, second-order model and constraints.

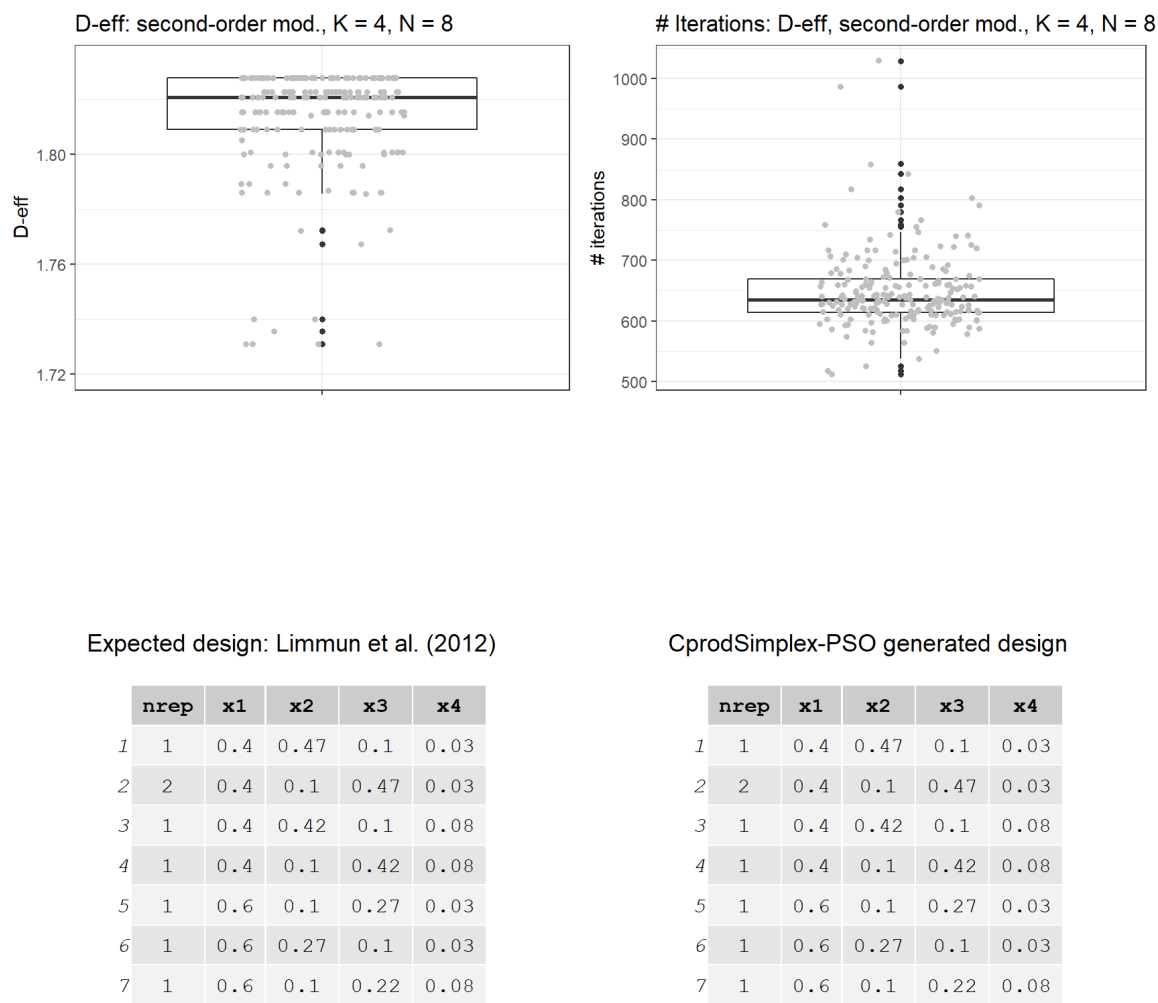


Figure 6.21: Results of the CprodSimplex-PSO search: $K = 4$, $N = 8$, first-order model and constraints.

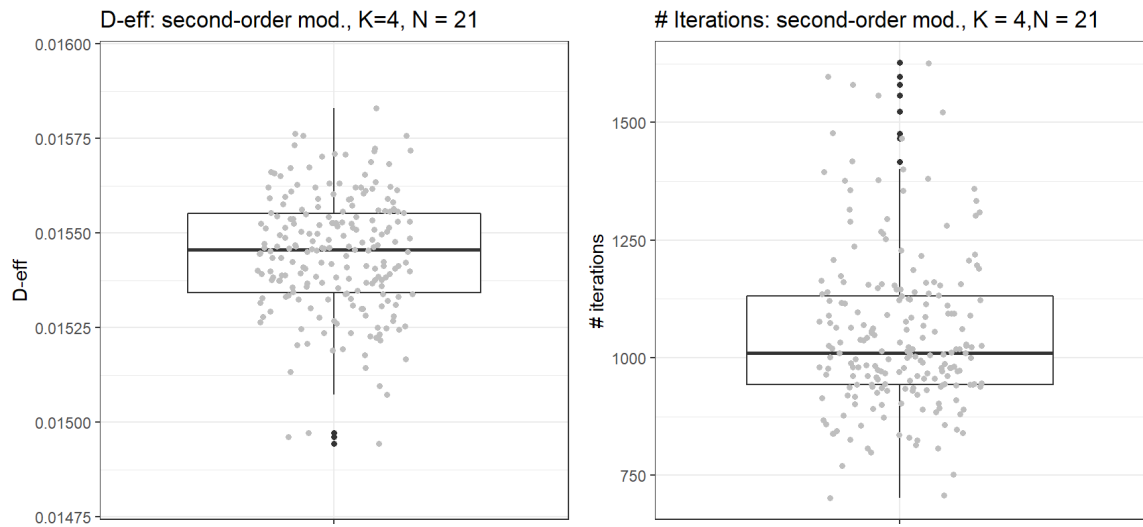


Figure 6.22: Results of the CprodSimplex-PSO search: $K = 4$, $N = 21$, second-order model and constraints.

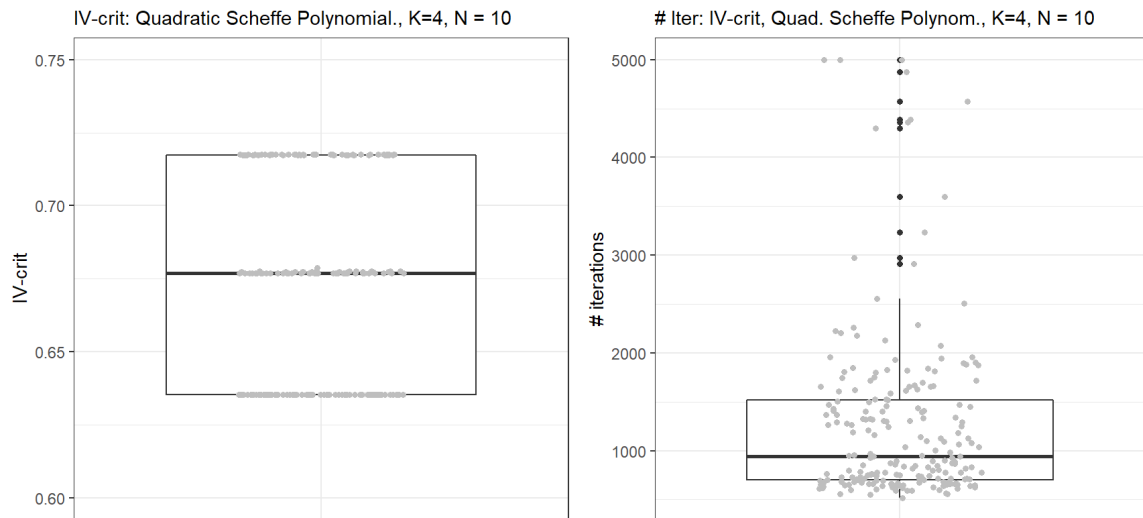


Figure 6.23: Results of the CprodSimplex-PSO search: $K = 4$, $N = 10$, quadratic Scheffé polynomial and constraints.

Table 6.5: The $K = 4$ $N = 21$, second-order model, with constraints D -PSO generated design.

$D\text{-eff} = 0.015830$			
x_1	x_2	x_3	x_4
0.4000	0.2856	0.2844	0.0300
0.4000	0.1000	0.4200	0.0800
0.4000	0.1000	0.4700	0.0300
0.6000	0.1000	0.2700	0.0300
0.4000	0.1000	0.4700	0.0300
0.4000	0.2671	0.2803	0.0527
0.4000	0.1000	0.4200	0.0800
0.5023	0.1000	0.3440	0.0537
0.6000	0.1000	0.2200	0.0800
0.5114	0.1000	0.3086	0.0800
0.6000	0.2200	0.1000	0.0800
0.4000	0.4200	0.1000	0.0800
0.4923	0.3277	0.1000	0.0800
0.4000	0.4426	0.1000	0.0574
0.6000	0.2452	0.1000	0.0548
0.6000	0.2700	0.1000	0.0300
0.6000	0.1000	0.2700	0.0300
0.4000	0.2772	0.2428	0.0800
0.5130	0.3570	0.1000	0.0300
0.4000	0.4700	0.1000	0.0300
0.4000	0.4700	0.1000	0.0300

Table 6.6: The CprodSimplex-PSO generated IV -optimal design for $K = 4$, $N = 10$, second-order Scheffé model. IV -CH is the score reported by Coetzer and Haines (2017) and their design is shown in parentheses.

$IV\text{-CH} = 0.6233$, $IV\text{-PSO} = 0.6237$					
Index	x_1	x_2	x_3	x_4	discrepancy
1	0.2 (0.2)	0.55 (0.55)	0.1 (0.1)	0.15 (0.15)	
2	0.2 (0.2)	0.393 (0.393)	0.153 (0.153)	0.254 (0.254)	
3	0.2 (0.2)	0.45 (0.45)	0.2 (0.2)	0.15 (0.15)	
4	0.228 (0.228)	0.222 (0.223)	0.2 (0.2)	0.35 (0.35)	
5	0.495 (0.495)	0.1 (0.1)	0.199 (0.199)	0.206 (0.206)	
6	0.439 (0.439)	0.1 (0.1)	0.135 (0.135)	0.327 (0.327)	
7	0.398 (0.398)	0.268 (0.268)	0.1 (0.1)	0.235 (0.235)	
8	0.65 (0.65)	0.1 (0.1)	0.1 (0.1)	0.15 (0.15)	
9	0.42 (0.42)	0.282 (0.282)	0.149 (0.149)	0.15 (0.15)	
10	0.2 (0.2)	0.35 (0.35)	0.1 (0.1)	0.35 (0.35)	

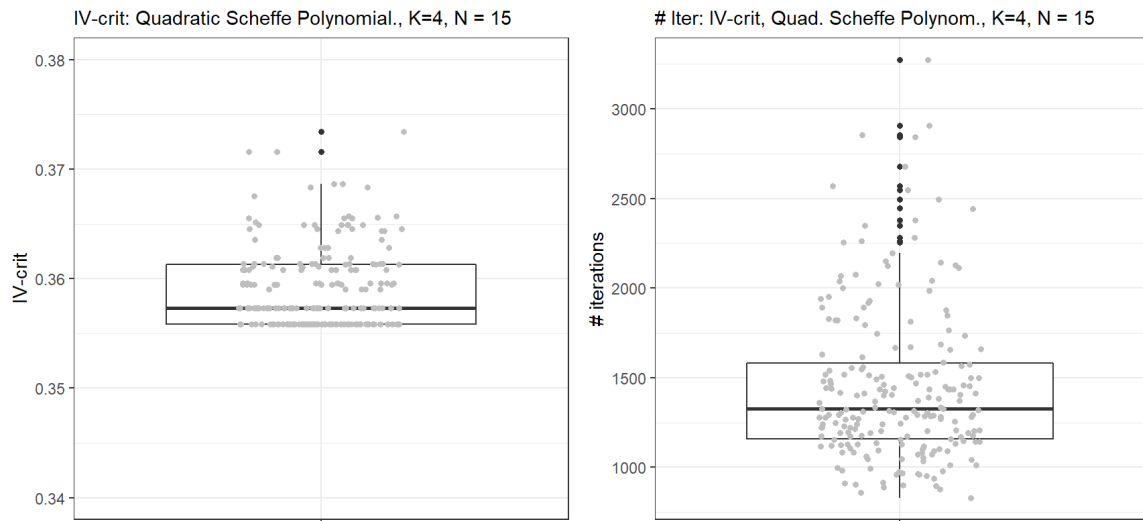


Figure 6.24: Results of the CprodSimplex-PSO search: $K = 4$, $N = 15$, quadratic Scheffé polynomial and constraints.

Table 6.7: The CprodSimplex-PSO generated *IV*-optimal design for $K = 4$, $N = 15$, second-order Scheffé model. *IV*-CH is the score reported by Coetzer and Haines (2017) and their design is shown in parentheses.

IV-CH = 0.345, IV-PSO = 0.3532, IV-releff = 98%					
index	x_1	x_2	x_3	x_4	discrepancy
1	0.55 (0.55)	0.1 (0.1)	0.2 (0.2)	0.15 (0.15)	
2	0.2 (0.2)	0.35 (0.35)	0.1 (0.1)	0.35 (0.35)	
3	0.2 (0.2)	0.45 (0.45)	0.2 (0.2)	0.15 (0.15)	
4	0.2 (0.2)	0.55 (0.55)	0.1 (0.1)	0.15 (0.15)	
5	0.397 (0.374)	0.266 (0.285)	0.1 (0.1)	0.238 (0.241)	*
6	0.65 (0.637)	0.1 (0.1)	0.1 (0.1135)	0.15 (0.15)	*
7	0.406 (0.399)	0.298 (0.297)	0.146 (0.154)	0.15 (0.15)	*
8	0.403 (0.424)	0.1 (0.1)	0.147 (0.126)	0.35 (0.35)	*
9	0.406 (0.432)	0.298 (0.318)	0.146 (0.1)	0.15 (0.15)	*
10	0.488 (0.438)	0.1 (0.1)	0.152 (0.2)	0.261 (0.262)	*
11	0.41 (0.508)	0.127 (0.1)	0.2 (0.1)	0.263 (0.292)	*
12	0.218 (0.2)	0.232 (0.25)	0.2 (0.2)	0.35 (0.35)	*
13	0.2 (0.2)	0.389 (0.407)	0.153 (0.149)	0.258 (0.25)	*
14	0.2 (0.323)	0.389 (0.285)	0.153 (0.152)	0.258 (0.24)	*
15	0.397 (0.323)	0.266 (0.285)	0.1 (0.152)	0.238 (0.24)	*

Table 6.8: Summary table of the performance of the new CprodSimplex-PSO algorithm to reproduce published D and IV -optimal designs for mixture experiments.

criterion	model	K	N	ground truth model	CprodSimplex-PSO result	median iterations	comment
D -criterion	first-order	3	3	$\{3, 1\}$	design produced exactly	24	all PSO runs found optimum
	second-order	3	6	$\{3, 2\}$	design produced exactly	170	all PSO runs found optimum
	first-order	4	4	$\{4, 1\}$	design produced exactly	30	all PSO runs found optimum
	second-order	4	10	$\{4, 2\}$	design produced exactly	320	all PSO runs found optimum
	special cubic	3	7	$\{3, 3\}$ -SCD	design produced exactly	230	all PSO runs found optimum
	special cubic	4	14	$\{4, 3\}$ -SCD	design produced to 3 decimal places	520	all PSO runs found optimum
	full cubic	3	10	augmented $\{4, 3\}$ -SCD	design produced exactly	330	all PSO runs found optimum
	full cubic	4	20	augmented $\{4, 4\}$ -SCD	design produced exactly	720	all PSO runs found optimum
IV -criterion	second-order	3	6	Goos et al. (2016)	slightly better IV -score found	200	all PSO runs found optimum
	second-order	3	7	Goos et al. (2016)	design produced exactly	220	all PSO runs found optimum
	second-order	3	8	Goos et al. (2016)	design produced exactly	255	all PSO runs found optimum
	second-order	3	30	Goos et al. (2016)	design produced exactly	665	all PSO runs found optimum
	second-order	4	15	Goos et al. (2016)	design produced exactly	580	all PSO runs found optimum
	special-cubic	4	16	Goos et al. (2016)	design produced to 3 decimal places	685	all PSO runs found optimum
	special-cubic	4	17	Goos et al. (2016)	design produced to 3 decimal places	935	all PSO runs found optimum

CHAPTER SEVEN

DISCUSSION, CONCLUSIONS, AND FUTURE WORK

7.1 Summary

In this dissertation we have provided a detailed and thorough development of extending and implementing Particle Swarm Optimization as an engine for generating exact optimal experimental designs. As we are bringing together topics from the machine learning and statistical design of experiments disciplines, we decided to include two introductory chapters, Chapters 1 and 2, which discuss, respectively, brief literature reviews on PSO and optimal DoE.

In Chapter 3 we extended the PSO to work as an optimizer for functions with matrix inputs, described the development of a `Julia` software package for implementing optimal design searches on the hypercube, and conducted a thorough benchmarking and validation study on three particle swarm variants. The validation shows a successful implementation of the algorithms and supports SPSO 2007 (which uses the local neighborhood communication topology) as a preferred algorithm for generating optimal DoEs.

In Chapter 4 we demonstrated the ability to produce new G -optimal designs that were yet unknown to the statistics community. Further, PSO is able to do this in a highly effective and efficient (w.r.t. computing speed) way.

In Chapter 5 we address a problem currently being discussed in the statistics literature: pure experimental replication is preferred for estimating the residual error variance which propagates through the entirety of the post-experiment linear-model analysis, and there is yet no optimal design algorithm that allows the user to specify an experimental replication structure while letting the optimal values of these

replicates be a parameter that varies during the optimality search. Our approach to generating optimal designs that contain a user specified replication structure relies on a redefinition of the model information matrix whereby the user can specify a general replication structure. The algorithm is guaranteed to ensure a minimum specified degrees-of-freedom for estimating the pure error variance. We believe this approach to be novel and quite promising and look forward to advancing this dimension of the work in future research.

Last, in Chapter 6 we tackled the problem of adapting PSO to generating optimal designs for mixture experiments. In this setting the design space is a standard $K - 1$ -simplex (i.e., the components of a design vector sum to 1). With respect to the optimization problem, the space of candidate design matrices is complicated as it is the Cartesian product of N standard $K - 1$ -simplices. The structure of this space created a number of difficulties in working with the standard Euclidean geometry to enact PSO. Therefore we adopted the Aitchison geometry, which requires redefinition of vector addition and scalar multiplication, but as it is demonstrated to be a vector and metric space it gave us all the tools required to develop a new PSO algorithm accomplishing this task. Our proposed algorithm was demonstrated to perform extremely well in reproducing very recent ‘currently best known optimal designs’ from papers dating to 2012, 2016, and 2018.

7.2 Conclusions

Based on the knowledge and intuition I have gained in working deeply with PSO in this research the last two years, I believe there is a real opportunity for PSO to become a strong and widely used toolbox for researchers and practitioners of experimental design. While there are some ‘heavy’ technical details that need fair attention when implementing PSO, the core intelligence of the algorithm is elegant

and quite simple. The algorithm makes no assumptions regarding the nature of the objective function. Further, it is highly robust to entrapment in local optima and specifically seeks the global optima.

While PSO has become amazingly successful in other scientific domains, it is quite interesting to note that its applications in the professional statistics sphere seem yet quite limited. Regarding optimal design of experiments, there are only a small handful of publications, which date back to 2011, illustrating PSO's efficacy. The majority of applications, heretofore, have been on generating optimal continuous designs and this dissertation fills some very large gaps in implementing the approach to generating optimal exact designs.

7.3 Planned future work

To motivate the future work, I end with a brief contrast of PSO's perspective on generating optimal designs with the current state-of-practice and 'go-to' algorithm: the coordinate exchange. The descriptions of these algorithms are of course only heuristic and extremely brief, but I intend the comparison here to highlight the difference in 'intelligence' (e.g., PSO is an agent-based approach that often comes with the phrase 'swarm intelligence') between the two algorithms. The coordinate exchange can involve a tremendous amount of detail for more difficult optimization problems (this is not adequately reflected in the description below).

Consider an optimal experimental design problem on K factors and denote a design point and design space as $\mathbf{x}' \in \mathcal{X} = [-1, 1]^K$. For exact N -point designs, the space of candidate design matrices, and thus the space we are searching with respect to some optimality property, is $\mathbf{X} \in \mathcal{X}^N$.

According to Goos and Jones (2011) [44] the basic structure of the coordinate-exchange algorithm is the following:

1. Draw a single candidate design randomly from the space of candidate design matrices: $\mathbf{X} \sim \mathcal{X}^N$.
2. Optimize locally: Loop through the NK rows and columns of $\mathbf{X}$ each time exchanging the current coordinate with both endpoints of that dimension, and if the exchange results in a better fitness in the objective, retain the exchanged coordinate.
 - (a) Repeat until no further improvement to the fitness criterion is found.
3. Retain the resulting matrix as a candidate for the optimal design. This is a locally optimum design, but there is no guarantee that there is not a better design that is not in this neighborhood.
4. Repeat steps 1-3 a recommended minimum of 1000 times (save for the easiest of optimality searches).
5. Among all generated candidates, the one with the best fitness score is retained and implemented.

In contrast a PSO based search takes the following approach:

1. Draw a small to moderately sized set of S candidate design matrices: $\{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_S\} \sim \mathcal{X}^N$
2. Seek global optimum of objective:
 - (a) Move, evaluate, remember, communicate.
 - (b) Converge to a group consensus on the location of the global optimizer.
3. Repeat independent PSO searches to increase chance of finding global optimizer.

The primary difference between the two algorithms, and the primary reason for PSOs efficiency as an approach to optimization, can be seen by comparing Step 1 of both algorithms. PSO is able to generate a more full perspective of the objective function over the space of candidate matrices from Step 1. This, in conjunction with the promising results presented in this dissertation, motivate me to continue this research path. I believe this approach will lead into searching for optimal designs on spaces and dimensions that have heretofore been difficult to study. The next steps include exploring further and expanding on the concepts shown in Chapters 5 and 6. Further, direct comparisons of PSO, coordinate exchange, and other machine learning algorithms for generating optimal designs are planned.

End.

REFERENCES CITED

- [1] J. Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (methodological)*, 44(2):139–177, 1982.
- [2] J. Aitchison. A general class of distributions on the simplex. *Journal of the Royal Statistical Society: Series B (methodological)*, 47(1):136–146, 1984.
- [3] J. Aitchison. Logratios and natural laws in compositional data analysis. *Mathematical Geology*, 31:563–580, 1999.
- [4] J. Aitchison and J. Bacon-Shone. Log contrast models for experiments with mixtures. *Biometrika*, 71(2):323–330, 1984.
- [5] A. Atkinson and R. Bailey. One hundred years of the design of experiments on and off the pages of biometrika. *Biometrika*, 88(1):53–97, 2001.
- [6] A. Atkinson, A. Donev, and R. Tobias. *Optimal Experimental Designs, with SAS*. Oxford Statistical Science Series, 2007.
- [7] L. Bain and M. Engelhardt. *Introduction to probability and mathematical statistics: second edition*. PWS-KENT Publishing Company, 1992.
- [8] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.
- [9] J. Bezanson, S. Karpinski, V. B. Shah, and A. Edelman. Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145*, 2012.
- [10] R. Boik. *Course Notes: Statistics 505 Linear Models*. Department of Mathematical Sciences, Montana State University, 2005.
- [11] R. Boik. *Course Notes: Statistics 550 Advanced Mathematical Statistics*. Department of Mathematical Sciences, Montana State University, 2018.
- [12] D. Boos and L. Stefanski. *Essential Statistical Inference: Theory and Methods*. Springer, 2013.
- [13] J. Borkowski. Using a genetic algorithm to generate small exact response surface designs. *Journal of Probability and Statistical Science*, 1(1), Feb. 2003.
- [14] J. Borkowski. *Course Notes: Statistics 578: Response Surface Methodology*. Department of Mathematical Sciences, Montana State University, 2019.
- [15] J. J. Borkowski. *GRAPHICAL METHODS FOR ASSESSING THE PREDICTION CAPABILITY OF RESPONSE SURFACE DESIGNS*, pages 349–378.

- [16] J. J. Borkowski. Finding maximum g-criterion values for central composite designs on the hypercube. *Communications in Statistics - Theory and Methods*, 24(8):2041–2058, 1995.
- [17] J. J. Borkowski. A comparison of prediction variance criteria for response surface designs. *Journal of Quality Technology*, 35(1):70–77, 2003.
- [18] J. J. Borkowski. *Center Points*. American Cancer Society, 2014.
- [19] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE Swarm Intelligence Symposium*, pages 120–127, April 2007.
- [20] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. *2007 IEEE Swarm Intelligence Symposium*, pages 120–127, 2007.
- [21] L. C. Cagnina, S. C. Esquivel, and C. A. C. Coello. Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. *Engineering Optimization*, 43(8):843–866, 2011.
- [22] G. Casella and R. L. Berger. *Statistical Inference: Second Edition*. Duxbury group., 2002.
- [23] R. Chen, S.-P. Chang, W. Wang, and W. Wong. Optimal experimental designs via particle swarm optimization methods. Technical report, Department of Mathematics, National Taiwan University, 2011.
- [24] R.-B. Chen, S.-P. Chang, W. Wang, H.-C. Tung, and W. K. Wong. Minimax optimal designs via particle swarm optimization methods. *Statistics and Computing*, 25(5):975–988, Sep 2015.
- [25] R.-B. Chen, D.-N. Hsieh, Y. Hung, and W. Wang. Optimizing latin hypercube designs by particle swarm. *Statistics and Computing*, 23(5):663–676, Sep 2013.
- [26] R.-B. Chen, Y.-W. Hsu, Y. Hung, and W. Wang. Discrete particle swarm optimization for constructing uniform design on irregular regions. *Computational Statistics and Data Analysis*, 72:282–297, 2014.
- [27] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1951–1957 Vol. 3, July 1999.
- [28] M. Clerc. Standard particle swarm optimisation. Technical report, HAL archives-ouvertes, 2012.
- [29] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, Feb 2002.

- [30] R. Coetzer and L. Haines. The construction of d - and i -optimal designs for mixture experiments with linear constraints on the components. *Chemometrics and Intelligent Laboratory Systems*, 171:112–124, 2017.
- [31] J. Cornell. *A primer on experiments with mixtures*. John Wiley and Sons, Inc., 2011.
- [32] D. Cox. *Planning of Experiments*. Wiley, New York., 1958.
- [33] R. B. Crosier. Mixture experiments: Geometry and pseudocomponents. *Technometrics*, 26(3):209–216, 1984.
- [34] R. B. Crosier. The geometry of constrained mixture experiments. *Technometrics*, 28(2):95–102, 1986.
- [35] N. Draper and A. Herzberg. On lack of fit. *Technometrics*, 13(2):231–241, 1971.
- [36] Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, volume 1, pages 81–86 vol. 1, May 2001.
- [37] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Oct 1995.
- [38] R. C. Eberhart and Y. Shi. Comparison between genetic algorithms and particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, pages 611–616, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
- [39] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 1, pages 84–88 vol.1, July 2000.
- [40] A. P. Engelbrecht. Particle swarm optimization: Global best or local best? In *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*, pages 124–135, Sep. 2013.
- [41] K.-T. Fang, Z. Geng, and G.-L. Tian. Statistical inference for truncated dirichlet distribution and its application in misclassification. *Biometrical Journal*, 42(8):1053–1068, 2000.
- [42] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian data analysis: Second edition*. Chapman and Hall/CRC, 2004.

- [43] S. Gilmour and L. Trinca. Optimum design of experiments for statistical inference. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(3):324–345, 2012.
- [44] P. Goos and B. Jones. *Optimal Design of Experiments: A Case study approach*. John Wiley and Sons, Ltd., 2011.
- [45] P. Goos, B. Jones, and U. Syafitri. I-optimal design of mixture experiments. *Journal of the American Statistical Association*, 111(514):899–911, 2016.
- [46] N. E. Hamilton and M. Ferry. ggtern: Ternary diagrams using ggplot2. *Journal of Statistical Software, Code Snippets*, 87(3):1–17, 2018.
- [47] L. N. Hernandez and C. J. Nachtsheim. Fast computation of exact G-optimal designs via λ -optimality. *Technometrics*, 60(3):297–305, 2018.
- [48] C. Heyde and E. Seneta, editors. *Statisticians of the Centuries*. Springer, 2001.
- [49] X. Hu and R. Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, pages 203–206, 2002.
- [50] X. Hu, R. Eberhart, and Y. Shi. Engineering optimization with particle swarm. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. (SIS’03)*, pages 53–57, 2003.
- [51] W. A. Jensen. Open problems and issues in optimal design. *Quality Engineering*, 30(4):583–593, 2018.
- [52] M. Jiao and J. Tang. A novel particle swarm optimization for constrained engineering optimization problems. In L. Kang, Z. Cai, X. Yan, and Y. Liu, editors, *Advances in Computation and Intelligence*, pages 79–88, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- [53] B. Jones and D. C. Montgomery. Partial replication of small two-level factorial designs. *Quality Engineering*, 29(2):190–195, 2017.
- [54] J. Kennedy. The particle swarm: social adaptation of knowledge. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*, pages 303–308, April 1997.
- [55] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1931–1938 Vol. 3, July 1999.
- [56] J. Kennedy. *Swarm Intelligence*, pages 187–220. Springer, 2006.

- [57] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.
- [58] J. Kiefer. Optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, 21(2):272–319, 1959.
- [59] J. Kiefer. Optimum designs in regression problems, ii. *The Annals of Mathematical Statistics*, 32(1):298–325, 1961.
- [60] Kutner, Nachtsheim, Neter, and Li. *Applied Linear Statistical Models. 5th Edition*. McGraw Hill., 2005.
- [61] W. Limmun, J. Borkowski, and B. Chomtee. Using a genetic algorithm to generate d -optimal designs for mixture experiments. *Quality and reliability engineering international*, 35(8):2657–2676, 2019.
- [62] J. Lukemire, A. Mandal, and W. K. Wong. d-qpso: A quantum-behaved particle swarm technique for finding d -optimal designs with discrete and continuous factors and a binary response. *Technometrics*, 61(1):77–87, 2019.
- [63] J. Marin, K. Mengersen, and C. Robert. *Bayesian modelling and inference on mixtures of distributions*, volume 25, pages 459–507. Elsevier, 2005.
- [64] K. Masuda, K. Kurihara, and E. Aiyoshi. A penalty approach to handle inequality constraints in particle swarm optimization. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2520–2525, 2010.
- [65] R. A. McLean and V. L. Anderson. Extreme vertices design of mixture experiments. *Technometrics*, 8(3):447–454, 1966.
- [66] R. K. Meyer and C. J. Nachtsheim. The coordinate-exchange algorithm for constructing exact optimal experimental designs. *Technometrics*, 37(1):60–69, 1995.
- [67] M. Morris. *Design of Experiments: An Introduction Based on Linear Models*. Chapman and Hall/CRC: Taylor and Francis group., 2017.
- [68] R. Myers, D. Montgomery, and C. Anderson-Cook. *Response surface methodology: process and product optimization using Designed Experiments. 4th Edition*. John Wiley and Sons, Ltd., 2016.
- [69] K. W. Ng, G.-L. Tian, and M.-L. Tang. *Truncated Dirichlet distribution*. John Wiley and Sons, Ltd., 2011.
- [70] A. K. Nigam, S. C. Gupta, and S. Gupta. A new algorithm for extreme vertices designs for linear mixture models. *Technometrics*, 25(4):367–371, 1983.

- [71] A. K. Nigam, S. C. Gupta, and S. Gupta. A new algorithm for extreme vertices designs for linear mixture models. *Technometrics*, 25(4):367–371, 1983.
- [72] K. Parsopoulos and M. Vrahatis. *Particle swarm optimization method for constrained optimization problems*. IOS Press, 2002.
- [73] G. F. Piepel. Defining consistent constraint regions in mixture experiments. *Technometrics*, 25(1):97–101, 1983.
- [74] G. F. Piepel, S. K. Cooley, and B. Jones. Construction of a 21-component layered mixture experiment design using a new mixture coordinate-exchange algorithm. *Quality Engineering*, 17(4):579–594, 2005.
- [75] V. Powlowsky-Glahn, J. Egozcue, and R. Tolosana-delgado. *The Aitchison geometry*. John Wiley and Sons, Ltd, 2015.
- [76] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019.
- [77] J. Rada-Vilela and M. Zhang. A performance study on synchronous and asynchronous updates in particle swarm optimization. In *GECCO '11 Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 21–28, July 2011.
- [78] M. Rodríguez, B. Jones, C. M. Borrór, and D. C. Montgomery. Generating and assessing exact g-optimal designs. *Journal of Quality Technology*, 42(1):3–20, 2010.
- [79] H. L. Royden. *Real Analysis second edition*. The Macmillan Company, 1968.
- [80] D. Sanders and C. Hild. A graphical tool for detection of outliers in completely randomized, unreplicated 2^k and 2^{k-p} factorials. *Quality Engineering*, 24(4):514–521, 2012.
- [81] H. Scheffé. Experiments with mixtures. *Journal of the Royal Statistical Society: Series B (methodological)*, 20(2):344–360, 1958.
- [82] H. Scheffé. The simple-centroid design for experiments with mixtures. *Journal of the Royal Statistical Society: Series B (methodological)*, 25(2):235–251, 1963.
- [83] C. Seber and C. Wild. *Nonlinear Regression*. Wiley, 2003.
- [84] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73, May 1998.

- [85] Y. Shi and R. C. Eberhart. Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, editors, *Evolutionary Programming VII*, pages 591–600, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.
- [86] Y. Shi, Z. Zhang, and W. Wong. Particle swarm based algorithms for finding locally and bayesian d-optimal designs. *Journal of Statistical Distributions and Applications*, 6(3), 04 2019.
- [87] K. Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards the proper choice of the distributions of observations. *Biometrika*, XII(1 and 2), 1918.
- [88] P. Srisuradetchai. *Robust response surface designs against missing observations*. Ph.D. Thesis. Montana State University., 2015.
- [89] W. Stein et al. *Sage Mathematics Software (Version x.y.z)*. The Sage Development Team, 2019. <http://www.sagemath.org>.
- [90] T. Takahama and S. Sakai. Constrained optimization by ϵ constrained particle swarm optimizer with ϵ -level control. In A. Abraham, Y. Dote, T. Furuhashi, M. Köppen, A. Ohuchi, and Y. Ohsawa, editors, *Soft Computing as Transdisciplinary Science and Technology*, pages 1019–1029, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [91] H. Uranisi. Optimal design for the special cubic regression model on the q -simplex. *Mathematical report 1: Kyushu University, General education department*, pages 902–909, 1964.
- [92] M. Urquhart, E. Ljungskog, and S. Sebben. Surrogate-based optimisation using adaptively scaled radial basis functions. *Applied and Soft Computing Journal*, 42, 2020.
- [93] B. Wheeler. *AlgDesign: Algorithmic Experimental Design*, 2019. R package version 1.2.0.
- [94] W. K. Wong, R.-B. Chen, C.-C. Huang, and W. Wang. A modified particle swarm optimization technique for finding optimal designs for mixture models. *PLOS ONE*, 10(6):1–23, 06 2015.
- [95] D. Wu and Hao Gao. Study on asynchronous update mechanism in particle swarm optimization. In *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pages 90–93, Sep. 2014.

- [96] Zambrano-Bigiarini, M., Rojas, and R. A model-independent particle swarm optimisation software for model calibration. *Environmental Modelling and Software*, 43:5–25, 2013.
- [97] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas-Mujica. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. *2013 IEEE Congress on Evolutionary Computation*, pages 2337–2344, 2013.
- [98] M. Zambrano-Bigiarini and R. Rojas. *hydroPSO: Particle Swarm Optimisation, with Focus on Environmental Models*, 2018. R package version 0.4-1 . doi:10.5281/zenodo.1287350.